Constant Factor Approximation for ATSP with Two Edge Weights
(Extended Abstract)

Ola Svensson[1], Jakub Tarnawski[1(✉)], and László A. Végh[2]

[1] École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
{ola.svensson,jakub.tarnawski}@epfl.ch
[2] London School of Economics, London, UK
L.Vegh@lse.ac.uk

Abstract. We give a constant factor approximation algorithm for the Asymmetric Traveling Salesman Problem on shortest path metrics of directed graphs with two different edge weights. For the case of unit edge weights, the first constant factor approximation was given recently in [17]. This was accomplished by introducing an easier problem called Local-Connectivity ATSP and showing that a good solution to this problem can be used to obtain a constant factor approximation for ATSP. In this paper, we solve Local-Connectivity ATSP for two different edge weights. The solution is based on a flow decomposition theorem for solutions of the Held-Karp relaxation, which may be of independent interest.

1 Introduction

The traveling salesman problem — one of finding the shortest tour of n cities — is one of the most classical optimization problems. Its definition dates back to the 19th century and since then a large body of work has been devoted to designing "good" algorithms using heuristics, mathematical programming techniques, and approximation algorithms. The focus of this work is on approximation algorithms. A natural and necessary assumption in this line of work that we also make throughout this paper is that the distances satisfy the triangle inequality: for any triple i, j, k of cities, we have $d(i, j)+d(j, k) \geq d(i, k)$ where $d(\cdot, \cdot)$ denotes the pairwise distances between cities. In other words, it is not more expensive to take the direct path compared to a path that makes a detour.

With this assumption, the approximability of TSP turns out to be a very delicate question that has attracted significant research efforts. Specifically, one of the first approximation algorithms (Christofides' heuristic [6]) was designed for the *symmetric* traveling salesman problem (STSP) where we assume symmetric distances ($d(i, j) = d(j, i)$); and, more recently, several works (see e.g. [1,3,8,9,17])

Please refer to the full version (http://arxiv.org/abs/1511.07038) for proofs and more detailed explanations (with figures).

O. Svensson—Supported by ERC Starting Grant 335288-OptApprox.
L.A. Végh—Supported by EPSRC First Grant EP/M02797X/1.

Q. Louveaux and M. Skutella (Eds.): IPCO 2016, LNCS 9682, pp. 226–237, 2016.
DOI: 10.1007/978-3-319-33461-5_19

have addressed the more general *asymmetric* traveling salesman problem (ATSP) where we make no such assumption.

However, there are still large gaps in our understanding of both STSP and ATSP. In fact, for STSP, the best approximation algorithm remains Christofides' 3/2-approximation algorithm from the 70's [6]. For the harder ATSP, the state of the art is a $\mathcal{O}(\log n / \log \log n)$-approximation algorithm by Asadpour et al. [3] and a recent $\mathcal{O}(\text{poly} \log \log n)$-estimation algorithm[1] by Anari and Oveis Gharan [1]. On the negative side, the best inapproximability results only say that STSP and ATSP are hard to approximate within factors 123/122 and 75/74, respectively [12]. Closing these gaps is a major open problem in the field of approximation algorithms (see e.g. "Problem 1" and "Problem 2" in the list of open problems in the recent book by Williamson and Shmoys [18]). What is perhaps even more intriguing about these questions is that we expect that a standard linear programming (LP) relaxation, often referred to as the Held-Karp relaxation, already gives better guarantees. Indeed, it is conjectured to give a guarantee of 4/3 for STSP and a guarantee of $\mathcal{O}(1)$ (or even 2) for ATSP.

An equivalent formulation of STSP and ATSP from a more graph-theoretic point of view is the following. For STSP, we are given a weighted undirected graph $G = (V, E, w)$ where $w : E \to \mathbb{R}_+$ and we wish to find a multisubset F of edges of minimum total weight such that (V, F) is connected and Eulerian. Recall that an undirected graph is Eulerian if every vertex has even degree. We also remark that we use the term multisubset as the solution F may use the same edge several times. An intuitive point of view on this definition is that G represents a road network, and a solution is a tour that visits each vertex at least once (and may use a single edge/road several times). The definition of ATSP is similar, with the differences that the input graph is directed and the output is Eulerian in the directed sense: the in-degree of each vertex equals its out-degree. Having defined the traveling salesman problem in this way, there are several natural special cases to consider. For example, what if G is planar? Or, what if all the edges/roads have the same length, i.e., if G is unweighted?

For planar graphs, we have much better algorithms than in general. Grigni et al. [11] first obtained a polynomial-time approximation scheme for STSP restricted to unweighted planar graphs, which was later generalized to edge-weighted planar graphs by Arora et al. [2]. More recently, ATSP on planar graphs (and more generally bounded genus graphs) was shown to admit constant factor approximation algorithms (first by Oveis Gharan and Saberi [9] and later by Erickson and Sidiropoulos [7] who improved the dependency on the genus).

In contrast to planar graphs, STSP and ATSP remain APX-hard for unweighted graphs (ones where all edges have identical weight) and, until recently, there were no better algorithms for these cases. Then, in a recent series of papers, the approximation guarantee of 3/2 was finally improved for STSP restricted to unweighted graphs. Specifically, Gharan et al. [10] first gave an

[1] An estimation algorithm is a polynomial-time algorithm for approximating/estimating the optimal value without necessarily finding a solution to the problem.

approximation guarantee of $1.5-\epsilon$; Mömke and Svensson [13] proposed a different approach yielding a 1.461-approximation guarantee; Mucha [14] gave a tighter analysis of this algorithm; and Sebő and Vygen [16] significantly developed the approach to give the currently best approximation guarantee of 1.4. Similarly, for ATSP, it was only very recently that the restriction to unweighted graphs could be leveraged: the first constant approximation guarantee for unweighted graphs was given by Svensson [17]. In this paper we make progress towards the general problem by addressing the simplest case left unresolved by [17]: graphs with two different edge weights.

Theorem 1.1. *There is an $\mathcal{O}(1)$-approximation algorithm for ATSP on graphs with two different edge weights.*

The paper [17] introduces an "easier" problem named Local-Connectivity ATSP, where one needs to find an Eulerian multiset of edges crossing only sets in a given partition rather than all possible sets (see next section for definitions). It is shown that an "α-light" algorithm to this problem yields a $(9+\varepsilon)\alpha$-factor approximation for ATSP. For unweighted graphs (and slightly more generally, for node-induced weight functions[2]) it is fairly easy to obtain a 3-light algorithm for Local-Connectivity ATSP; the difficult part in [17] is the black-box reduction of ATSP to this problem. Note that [17] easily gives an $\mathcal{O}(w_{\max}/w_{\min})$-approximation algorithm in general if we take $w_{\max}$ and $w_{\min}$ to denote the largest and smallest edge weight, respectively. However, obtaining a constant factor approximation even for two different weights requires substantial further work.

In Local-Connectivity ATSP we need a lower bound function $\mathrm{lb} : V \to \mathbb{R}_+$ on the vertices. The natural choice for node-induced weights is $\mathrm{lb}(v) = \sum_{e\in\delta^+(v)} w(e)x^*_e$. With this weight function, every vertex is able to "pay" for the incident edges in the Eulerian subgraph we are looking for. This choice of lb does not seem to work for more general weight functions, and we need to define lb more "globally", using a new flow theorem for Eulerian graphs (Theorem 2.4). In Sect. 1.2, after the preliminaries, we give a more detailed overview of these techniques and the proof of the theorem. Our argument is somewhat technical, but it demonstrates the potential of the Local-Connectivity ATSP problem as a tool for attacking general ATSP.

Finally, let us remark that both STSP [4,15] and ATSP [5] have been studied in the case when all distances are either 1 or 2. That restriction is very different from our setting, as in those cases the input graph is complete. In particular, it is trivial to get a 2-approximation algorithm there, whereas in our setting – where the input graph is *not* complete – a constant factor approximation guarantee already requires non-trivial algorithms.

[2] For ATSP, we can think of a node-weighted graph as an edge-weighted graph where the weight of an edge (u, v) equals the node weight of u.

1.1 Notation and Preliminaries

We consider an edge-weighted directed graph $G = (V, E, w)$ with $w : E \to \mathbb{R}_+$. For a vertex subset $S \subseteq V$ we let $\delta^+(S) = \{(u, v) \in E : u \in S, v \in V \setminus S\}$ and $\delta^-(S) = \{(u, v) \in E : u \in V \setminus S, v \in S\}$ denote the sets of outgoing and incoming edges, respectively. For a subset of edges $E' \subseteq E$, we use $\delta^+_{E'}(S) = \delta^+(S) \cap E'$ and $\delta^-_{E'}(S) = \delta^-(S) \cap E'$. We also let $\mathcal{C}(E') = (\tilde{G}_1, \ldots, \tilde{G}_k)$ denote the set of weakly connected components of the graph (V, E'); the vertex set V will always be clear from the context. For a directed graph $\tilde{G}$ we use $V(\tilde{G})$ to denote its vertex set and $E(\tilde{G})$ the edge set. For brevity, we denote the singleton set $\{v\}$ by v (e.g. $\delta^+(v) = \delta^+(\{v\})$), and we use the notation $x(F) = \sum_{e \in F} x_e$ for a subset $F \subseteq E$ of edges. For the case of two edge weights, we use $0 \leq w_0 < w_1$ to denote the two possible values, and partition $E = E_0 \cup E_1$ so that $w(e) = w_0$ if $e \in E_0$ and $w(e) = w_1$ if $e \in E_1$. We will refer to edges in E_0 and E_1 as cheap and expensive edges, respectively.

We define ATSP as the problem of finding a connected Eulerian subgraph of minimum weight. As already mentioned in the introduction, this definition is equivalent to that of visiting each city exactly once (in the metric completion) since we assume the triangle inequality. The formal definition is as follows.

ATSP

Given: An edge-weighted (strongly connected) digraph $G = (V, E, w)$.
Find: A multisubset F of E of minimum total weight $w(F) = \sum_{e \in F} w(e)$ such that (V, F) is Eulerian and connected.

Held-Karp Relaxation. The Held-Karp relaxation has a variable $x_e \geq 0$ for every edge in G. The intended meaning is that x_e should equal the number of times e is used in the solution. The relaxation $\mathrm{LP}(G)$ is defined as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} w(e) x_e \\ \text{subject to} \quad & x(\delta^+(v)) = x(\delta^-(v)) \qquad v \in V, \\ & x(\delta^+(S)) \geq 1 \qquad \emptyset \neq S \subsetneq V, \\ & x \geq 0. \end{aligned} \tag{LP(G)}$$

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. The second set of constraints enforces that the solution is connected; they are sometimes referred to as subtour elimination constraints. Finally, we remark that although the Held-Karp relaxation has exponentially many constraints, it is well-known that we can solve it in polynomial time either by using the ellipsoid method with a separation oracle or by formulating an equivalent compact (polynomial-size) linear program. We

will use x^* to denote an optimal solution to LP(G) of value OPT, which is a lower bound on the value of an optimal solution to ATSP on G.

Local-Connectivity ATSP. The Local-Connectivity ATSP problem can be seen as a two-stage procedure. In the first stage, the input is an edge-weighted digraph $G = (V, E, w)$ and the output is a "lower bound" function $\text{lb} : V \to \mathbb{R}_+$ on the vertices such that $\text{lb}(V) \leq \text{OPT}$. In the second stage, the input is a partition of the vertices, and the output is an Eulerian multisubset of edges which crosses each set in the partition and where the ratio of weight to lb of every connected component is as small as possible. We now give the formal description of the second stage, assuming the lb function is already computed.

Local-Connectivity ATSP

Given: An edge-weighted digraph $G = (V, E, w)$, a function $\text{lb} : V \to \mathbb{R}_+$ with $\text{lb}(V) \leq \text{OPT}$, and a partitioning $V = V_1 \cup V_2 \cup \ldots \cup V_k$ of the vertices.
Find: A Eulerian multisubset F of E such that

$$|\delta_F^+(V_i)| \geq 1 \quad \text{for } i = 1, 2, \ldots, k \qquad \text{and} \qquad \max_{\tilde{G} \in \mathcal{C}(F)} \frac{w(\tilde{G})}{\text{lb}(\tilde{G})} \text{ is minimized.}$$

Here we used the notation that for a connected component $\tilde{G}$ of (V, F), $w(\tilde{G}) = \sum_{e \in E(\tilde{G})} w(e)$ (summation over the edges) and $\text{lb}(\tilde{G}) = \sum_{v \in V(\tilde{G})} \text{lb}(v)$ (summation over the vertices). We say that an algorithm for Local-Connectivity ATSP is α-*light* on G if it is guaranteed, for any partition, to find a solution F such that for every component $\tilde{G} \in \mathcal{C}(F)$, $w(\tilde{G})/\text{lb}(\tilde{G}) \leq \alpha$.

In [17], lb is defined as $\text{lb}(v) = \sum_{e \in \delta^+(v)} w(e) x_e^*$; note that $\text{lb}(V) = OPT$ in this case. We remark that we use the "α-light" terminology to avoid any ambiguities with the concept of approximation algorithms (an α-light algorithm does not compare its solution to an optimal solution to the given instance of Local-Connectivity ATSP).

Perhaps the main difficulty of ATSP is to satisfy the connectivity requirement, i.e., to select an Eulerian subset F of edges which connects the whole graph. Local-Connectivity ATSP relaxes this condition – we only need to find an Eulerian set F that crosses the k cuts defined by the partition. This makes it intuitively an "easier" problem than ATSP. Indeed, an α-approximation algorithm for ATSP (with respect to the Held-Karp relaxation) is trivially an α-light algorithm for Local-Connectivity ATSP for an arbitrary lb function with $\text{lb}(V) = OPT$: just return the same Eulerian subset F as the algorithm for ATSP; since the set F connects the graph, we have $\max_{\tilde{G} \in \mathcal{C}(F)} w(\tilde{G})/\text{lb}(\tilde{G}) = w(F)/\text{lb}(V) \leq \alpha$. Perhaps more surprisingly, the main technical theorem of [17] shows that the two problems are equivalent up to small constant factors.

Theorem 1.2 [17]**.** *Let $\mathcal{A}$ be an algorithm for Local-Connectivity ATSP. Consider an ATSP instance $G = (V, E, w)$, and let* OPT *denote the optimum value of the Held-Karp relaxation. If $\mathcal{A}$ is α-light on G, then there exists a tour of G with value at most $5\alpha\,\mathrm{OPT}$. Moreover, for any $\varepsilon > 0$, a tour of value at most $(9+\varepsilon)\alpha\,\mathrm{OPT}$ can be found in time polynomial in the number $n = |V|$ of vertices, in $1/\varepsilon$, and in the running time of $\mathcal{A}$.*

In other words, the above theorem says that in order to approximate an ATSP instance G, it is sufficient to devise a polynomial-time algorithm to calculate a lower bound lb and a polynomial time algorithm for Local-Connectivity ATSP that is $\mathcal{O}(1)$-light on G with respect to this lb function. Our main result is proved using this framework.

1.2 Technical Overview

Singleton partition. Let us start by outlining the fundamental ideas of our algorithm and comparing it to [17] for the special case of Local-Connectivity ATSP when all partition classes V_i are singletons. For unit weights, the choice $\mathrm{lb}(v) = \sum_{e\in\delta^+(v)} w(e)x^\star_e = x^\star(\delta^+(v))$ in [17] is a natural one: intuitively, every node is able to pay for its outgoing edges. We can thus immediately give an algorithm for this case: just select an arbitrary integral solution z to the circulation problem with node capacities $1 \le z(\delta^+(v)) \le \lceil x^\star(\delta^+(v))\rceil$. Then for any v we have $z(\delta^+(v)) \le x^\star(\delta^+(v)) + 1 \le 2x^\star(\delta^+(v))$ and hence $\sum_{e\in\delta^+(v)} w(e)z_e \le 2\,\mathrm{lb}(v)$, showing that z is a 2-light solution.

The same choice of lb does not seem to work in the presence of two different edge costs. Consider a case when every expensive edge carries only a small fractional amount of flow. Then $\sum_{e\in\delta^+(v)} w(e)x^\star_e$ can be much smaller than the expensive edge cost w_1, and thus the vertex v would not be able to "afford" even a single outgoing expensive edge. To resolve this problem, we bundle small fractional amounts of expensive flow, channelling them to reach a small set of terminals. This is achieved via Theorem 2.4, a flow result which might be of independent interest. It shows that within the fractional Held-Karp solution $x^\star$, we can send the flow from an arbitrary edge set E' to a sink set T with $|T| \le 8x^\star(E')$; in fact, T can be any set minimal for inclusion such that it can receive the total flow from E'. We apply this theorem for $E' = E_1$, the set of expensive edges; let f be the flow from E_1 to T, and call elements of T *terminals*. Now, whenever an expensive edge is used, we will "force" it to follow f to a terminal in T, where it can be paid for. Enforcement is technically done by splitting the vertices into two copies, one carrying the f flow and the other the rest. Thus we obtain the *split graph* G_{sp} and split fractional optimal solution $x^\star_{\mathrm{sp}}$.

The design of the split graph is such that every walk in it which starts with an expensive edge must proceed through cheap edges until it reaches a terminal before visiting another expensive edge. In our terminology, expensive edges create "debt", which must be paid off at a terminal. Starting from an expensive edge, the debt must be carried until a terminal is reached, and no further debt

can be taken in the meantime. The bound on the number of terminals guarantees that we can assign a lower bound function lb with $\mathrm{lb}(V) \leq \mathrm{OPT}$ such that (up to a constant factor) cheap edges are paid for locally, at their heads, whereas expensive edges are paid for at the terminals they are routed to. Such a splitting easily solves Local-Connectivity ATSP for the singleton partition: find an arbitrary integral circulation z_{sp} in the split graph with an upper bound $z_{\mathrm{sp}}(\delta^+(v)) \leq \lceil 2x^\star_{\mathrm{sp}}(\delta^+(v)) \rceil$ on every node, and a lower bound 1 on whichever copy of v transmits more flow. Note that $2x^\star_{\mathrm{sp}}$ is a feasible fractional solution to this problem. We map z_{sp} back to an integral circulation z in the original graph by merging the split nodes, thus obtaining a constant-light solution.

Arbitrary partitions. Let us now turn to the general case of Local-Connectivity ATSP, where the input is an arbitrary partition $V = V_1 \cup \ldots \cup V_k$. For unit weights this is solved in [17] via an integer circulation problem on a modified graph. Namely, an auxiliary node A_i is added to represent each partition class V_i, and one unit of in- and outgoing flow from V_i is rerouted through A_i. In the circulation problem, we require exactly one in- and one outgoing edge incident to A_i to be selected. When we map the solution back to the original graph, there will be one incoming and one outgoing arc from every set V_i (thus satisfying the connectivity requirement) whose endpoints inside V_i violate the Eulerian condition. In [17] every V_i is assumed to be strongly connected, and therefore we can "patch up" the circulation by connecting the loose endpoints by an arbitrary path inside V_i. This argument easily gives a 3-light solution.

Let us observe that the strong connectivity assumption is in fact not needed for the result in [17]. Indeed, given a component V_i which is not strongly connected, consider its decomposition into strongly connected (sub)components, and pick a $U_i \subseteq V_i$ which is a sink (i.e. it has no edges outgoing to $V_i \setminus U_i$). We proceed by rerouting 1 unit of flow through a new auxiliary vertex just as in that algorithm, but we do this for U_i instead. This guarantees that U_i has at least one outgoing edge in our solution, and that edge must leave V_i as well.

We now turn to our result for two different edge weights. We are aiming for a similar construction as in the unit-weight case: based on the split graph G_{sp}, we construct an integer circulation problem with an auxiliary vertex A_i representing a certain subset $U_i \subseteq V_i$ for every $1 \leq i \leq k$. We then map its solution back to the original graph and patch up the loose endpoints inside every U_i by a path. However, we have to account for the following difficulties: *(i)* an edge leaving U_i should also leave V_i; *(ii)* debt should not disappear inside U_i: if the edge entering it carries debt but the edge leaving does not, we must make sure this difference can be charged to a terminal in U_i; *(iii)* the path used inside U_i must pay for all expensive edges it uses. All three issues can be appropriately tackled by defining an auxiliary graph inside V_i. Edges of the auxiliary graph represent paths containing one expensive edge and one terminal (which can pay for themselves); however, these paths may not map to paths in the split graph. We select the subset $U_i \subseteq V_i$ as a sink component in the auxiliary graph.

2 Algorithm for Local-Connectivity ATSP

We prove our main result in this section. Our claim for ATSP follows from solving Local-Connectivity ATSP:

Theorem 2.1. *There is a polynomial-time* 100*-light algorithm for Local-Connectivity ATSP on graphs with two edge weights.*

Together with Theorem 1.2, this implies our main result:

Theorem 2.2. *For any graph with two edge weights, the integrality gap of its Held-Karp relaxation is at most* 500*. Moreover, we can find an* 901*-approximate tour in polynomial time.*

The factor 500 comes from $5 \cdot 100$, and 901 is selected so that $(9+\varepsilon) \cdot 100 \leq 901$. Our proof of Theorem 2.1 proceeds as outlined in Sect. 1.2. In this extended abstract, we only describe the construction; the proof is given in the full version.

Recall that the edges are partitioned into the set E_0 of cheap edges and the set E_1 of expensive edges. Set $x^\star$ to be an optimal solution to the Held-Karp relaxation. We start by noting that the problem is easy if $x^\star$ assigns very small total fractional value to expensive edges. In that case, we can easily reduce the problem to the unweighted case which was solved in [17].

Lemma 2.3. *There is a polynomial-time* 6*-light algorithm for Local-Connectivity ATSP for graphs where* $x^\star(E_1) < 1$*.*

For the rest of this section, we thus assume $x^\star(E_1) \geq 1$. Our objective is to define a function $\mathrm{lb} : V \to \mathbb{R}_+$ such that $\mathrm{lb}(V) \leq \mathrm{OPT} = w(x^\star)$ and then show how to, given a partition $V = V_1 \cup \ldots \cup V_k$, find an Eulerian set of edges F which crosses all V_i-cuts and is $\mathcal{O}(1)$-light with respect to the defined lb function.

2.1 Calculating lb and Constructing the Split Graph

Finding terminals T and flow f. For this, we use the following flow result.

Theorem 2.4. *Let $D = (V \cup \{s\}, E)$ be a directed graph, $c : E \to \mathbb{R}_+$ – a nonnegative capacity vector, and s – a source node with no incoming edges, i.e., $\delta^-(s) = \emptyset$. Assume that for all $\emptyset \neq S \subseteq V$ we have*

$$c(\delta^-(S)) \geq \max\{1, c(\delta^+(S))\}. \tag{1}$$

Consider a set $T \subseteq V$ such that there exists a flow $f \leq c$ of value $c(\delta^+(s))$ from the source s to the sink set T, and T is minimal subject to this property. Then $|T| \leq 8c(\delta^+(s))$.

Corollary 2.5. *There exist a vertex set $T \subseteq V$ and a flow $f : E \to \mathbb{R}_+$ from source set $\{\mathrm{tail}(e) : e \in E_1\}$ to sink set T of value $x^\star(E_1)$ such that: (a) $|T| \leq 8x^\star(E_1)$, (b) $f \leq x^\star$, (c) f saturates all expensive edges, i.e., $f(e) = x^\star_e$ for all $e \in E_1$, (d) for each $t \in T$, $f(E_0 \cap \delta^+(t)) = 0$ and $f(\delta^-(t)) > 0$. Moreover, T and f can be computed in polynomial time.*

Definition of lb. We set $\mathrm{lb} : V \to \mathbb{R}_+$ to be a scaled-down variant of $\overline{\mathrm{lb}} : V \to \mathbb{R}_+$ which is defined as follows:

$$\overline{\mathrm{lb}}(v) := \begin{cases} w_0 \cdot x^\star(\delta^-(v)) & \text{if } v \notin T, \\ w_0 \cdot x^\star(\delta^-(v)) + w_1 \cdot \lceil f(\delta^-(t)) \rceil & \text{if } v \in T. \end{cases}$$

The definition of lb is now simply $\mathrm{lb}(v) = \overline{\mathrm{lb}}(v)/10$. The scaling-down is done so as to satisfy $\mathrm{lb}(V) \leq \mathrm{OPT}$ (see Lemma 2.6). Clearly we have $\overline{\mathrm{lb}}(v) \geq w_0$ for all $v \in V$ and $\overline{\mathrm{lb}}(t) \geq w_1 + w_0 \geq w_1$ for terminals $t \in T$.

The intuition behind this setting of $\overline{\mathrm{lb}}$ is that we want to pay for each expensive edge $e \in E_1$ in the terminal $t \in T$ which the flow f "assigns" to e. Indeed, in the split graph we will reroute flow (using f) so as to ensure that any path which traverses e must then visit such a terminal t to offset the cost of the expensive edge.

Lemma 2.6. $\overline{\mathrm{lb}}(V) \leq 10 \cdot \mathrm{OPT}$.

Construction of the split graph. The next step is to reroute flow so as to ensure that all expensive edges are "paid for" by the lb at terminals. To this end, we define a new *split graph* and a *split circulation* on it.

Definition 2.7. *The* split graph G_{sp} *is defined as follows. For every* $v \in V$ *we create two copies* v^0 *and* v^1 *in* $V(G_{\mathrm{sp}})$. *For every cheap edge* $(u, v) \in E_0$*:*

- *if* $x^\star(u, v) - f(u, v) > 0$, *create an edge* (u^0, v^0) *in* $E(G_{\mathrm{sp}})$ *with* $x^\star_{\mathrm{sp}}(u, v) = x^\star(u, v) - f(u, v)$,
- *if* $f(u, v) > 0$, *create an edge* (u^1, v^1) *in* $E(G_{\mathrm{sp}})$ *with* $x^\star_{\mathrm{sp}}(u, v) = f(u, v)$.

For every expensive edge $(u, v) \in E_1$ *we create one edge* (u^0, v^1) *in* $E(G_{\mathrm{sp}})$ *with* $x^\star_{\mathrm{sp}}(u, v) = f(u, v)$. *Finally, for each* $t \in T$ *we create an edge* (t^1, t^0) *in* $E(G_{\mathrm{sp}})$ *with* $x^\star_{\mathrm{sp}}(t^1, t^0) = f(\delta^-(t))$.

The new edges are weighted as follows: images of edges in E_0 *have weight* w_0, *the images of edges in* E_1 *have weight* w_1, *and the new edges* (t^1, t^0) *have weight* 0. *Let us denote the new weight function by* w_{sp}.

Vertices v^0 *will be called* free vertices *and vertices* v^1 *will be called* debt vertices. *Edges entering a free vertex will be called* free edges, *and those entering a debt vertex will be called* debt edges.

By construction we have that *(a)* $x^\star_{\mathrm{sp}}$ is a circulation on G_{sp}, *(b)* (the image of) every cut is still crossed by at least 1 unit of $x^\star_{\mathrm{sp}}$, and *(c)* any path in G_{sp} which begins with a debt edge and ends with a free edge must go through a terminal.

2.2 Solving Local-Connectivity ATSP

Now our algorithm is given a partition $V = V_1 \cup ... \cup V_k$. The objective is to output a set of edges F which crosses all V_i-cuts and is $\mathcal{O}(1)$-light with respect to our lb function. We do so by first defining *auxiliary graphs* that help us modify the split graph so as to force our solution to cross the cuts defined by the partition. We then use such a flow to define the set F of edges.

Construction of auxiliary graphs and modification of split graph. Our first step is to construct an *auxiliary graph* for each component V_i. The strong-connectivity structure of this graph will guide our algorithm.

Definition 2.8. *The* auxiliary graph G_i^{aux} *is a graph with vertex set* V_i *and the following edge set: for* $u, v \in V_i$, $(u, v) \in E(G_i^{\text{aux}})$ *if any of the following three conditions is satisfied:*

- *there is a cheap edge* $(u, v) \in E_0 \cap G[V_i]$ *inside* V_i, *or*
- *there is a* u*-*v*-path in* $G[V_i]$ *whose first edge is expensive and all other edges are cheap, and* $v \in T$ *is a terminal – we then call the edge* $(u, v) \in E(G_i^{\text{aux}})$ *a* postpaid edge *– or*
- *there is a* u*-*v*-path in* $G[V_i]$ *whose last edge is expensive and all other edges are cheap, and* $u \in T$ *is a terminal – we then call the edge* $(u, v) \in E(G_i^{\text{aux}})$ *a* prepaid edge.

Define the preimage *of such an edge* $(u, v) \in E(G_i^{\text{aux}})$ *to be the* shortest *path inside* V_i *as above (in the first case, a single edge).*

Now, for each i consider a decomposition of G_i^{aux} into strongly connected components. Let $U_i \subseteq V_i$ be the vertex set of a sink component in this decomposition. That is, there is no edge from U_i to $V_i \setminus U_i$ in the auxiliary graph G_i^{aux}. Note that G_i^{aux} is constructed based only on the original graph G and not the split graph G_{sp}. However, we will solve Local-Connectivity ATSP by solving an integral circulation problem on G'_{sp}: a modification of the split graph G_{sp}, described as follows.

For each i, define $U_i^{\text{sp}} = \{v^0, v^1 : v \in U_i\} \subseteq V(G_{\text{sp}})$ to be the set of vertices in the split graph corresponding to U_i. (Note that U_i^{sp} may not be strongly connected in G_{sp}.) We are going to reroute part of the $x^\star_{\text{sp}}$ flow going in and out of U_i^{sp} to a new auxiliary vertex A_i. While the 3-light algorithm for unit-weight graphs rerouted flow from all boundary edges of a component U_i (see Sect. 1.2), here we will be more careful and choose only a subset of boundary edges of U_i^{sp} to be rerouted.

To this end, select a subset of edges $X_i^- \subseteq \delta^-(U_i^{\text{sp}})$ with $x^\star_{\text{sp}}(X_i^-) = 1/2$ such that either all edges in X_i^- are debt edges, or all are free edges.

We define the set of outgoing edges $X_i^+ \subseteq \delta^+(U_i^{\text{sp}})$ to be, intuitively, the edges over which the flow that entered U_i^{sp} by X_i^- exits U_i^{sp}. That is, consider an arbitrary cycle decomposition of the circulation $x^\star_{\text{sp}}$, and look at the set of cycles containing the edges in X_i^-. We define X_i^+ as the set of edges on these cycles that first leave U_i^{sp} after entering U_i^{sp} on an edge in X_i^-; clearly, $x^\star_{\text{sp}}(X_i^+) = 1/2$.

Let g_i denote the flow on these cycles connecting the heads of edges in X_i^- and the tails of edges in X_i^+. We will use the following claim later in the construction.

Fact 2.9. *Assume all edges in* X_i^- *are debt edges but* $e \in X_i^+$ *is a free edge or an expensive edge. Then there exists a path in* $G_{\text{sp}}[U_i]$ *between a vertex* t^0 *(for some terminal* $t \in T$*) and the tail of* e*, made up of only cheap edges.*

We now transform G_{sp} into a new graph G'_{sp} and $x^\star_{\mathrm{sp}}$ into new circulation x'_{sp} as follows. For every set V_i in the partition 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 further subtract the flow g_i inside U_i^{sp}; hence the resulting vector x'_{sp} will be a circulation, with $x'_{\mathrm{sp}}(\delta^-(A_i)) = 1/2$. If X_i^- is a set of free edges, then we will say that A_i is a free vertex, otherwise we say that it is a debt vertex.

Transforming x'_{sp} into an integral flow and obtaining our solution F. In the next step we round x'_{sp} to integrality while respecting degrees of vertices:

Lemma 2.10. *There exists an integral circulation y'_{sp} on G'_{sp} satisfying the following conditions: (a) $y'_{\mathrm{sp}}(\delta^-(v)) \leq \lceil 2x^\star_{\mathrm{sp}}(\delta^-(v)) \rceil$ for each $v \in V(G_{\mathrm{sp}})$, (b) $y'_{\mathrm{sp}}(\delta^-(A_i)) = 1$ for each i. Such a circulation y'_{sp} can be found in polynomial time.*

We will now transform y'_{sp} into an Eulerian set of edges F in the original graph G. We can think of this as a three-stage process.

First, we map all edges adjacent to the auxiliary vertices A_i back to their preimages in G_{sp}, obtaining from y'_{sp} an integral *pseudo-flow* y_{sp} in G_{sp}. (We use the term *pseudo-flow* as now, some vertices may not satisfy flow conservation.)

Second, we contract the two copies v^0 and v^1 of every vertex $v \in V$, thus mapping all edges back to their preimages in G. (We remove all edges (t^1, t^0) for $t \in T$.) This creates an integral pseudo-flow y in G.

Since the in- and out-degree of A_i were exactly 1 in y'_{sp}, now (in y) 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 where flow conservation in y can be violated. As the third step, to repair this, we route a walk P_i from u_i to v_i. Our Eulerian set of edges $F \subseteq E$ which we finally return is the integral pseudo-flow y plus the union (over i) of all such walks P_i, i.e., $\mathbb{1}_F = y + \sum_i \mathbb{1}_{P_i}$.

It remains to describe how we route these paths. Fix i. Recall that U_i is strongly connected in G_i^{aux}. We distinguish two cases:

- If A_i is a free vertex or the edge exiting A_i in y'_{sp} (in G'_{sp}) is a debt edge, then select a shortest u_i-v_i-path in G_i^{aux}, map each edge of this path to its preimage path (see Definition 2.8) and concatenate them to obtain a u_i-v_i-walk P_i in V_i.
- If A_i is a debt vertex but the edge exiting A_i in y'_{sp} (in G'_{sp}) is a free edge, then by Fact 2.9 there is a terminal t inside U_i, with a path from t to v_i using only cheap edges. Proceed as above to obtain a u_i-t-walk and then append this cheap t-v_i-path to it, obtaining a u_i-v_i-walk P_i in V_i.

This concludes the description of the algorithm. In the full version of the paper we prove that the returned Eulerian set of edges F has the properties we desire, i.e.,

Lemma 2.11. *For every connected component $\widetilde{G}$ of (V, F) we have $w(\widetilde{G}) \leq 10 \cdot \overline{\mathrm{lb}}(\widetilde{G})$.*

Lemma 2.12. *For every component V_i we have $|\delta_F^+(V_i)| \geq 1$.*

Lemmas 2.6 and 2.11 together prove that our algorithm is 100-light with respect to lb.

References

1. Anari, N., Gharan, S.O.: Effective-resistance-reducing flows and asymmetric TSP. CoRR, abs/1411.4613 (2014)
2. Arora, S., Grigni, M., Karger, D.R., Klein, P.N., Woloszyn, A.: A polynomial-time approximation scheme for weighted planar graph TSP. In: Proceedings of SODA, vol. 98, pp. 33–41 (1998)
3. Asadpour, A., Goemans, M.X., Madry, A., Gharan, S.O., Saberi, A.: An O(log n/ log log n)-approximation algorithm for the asymmetric traveling salesman problem. In: Proceedings of SODA, pp. 379–389 (2010)
4. Berman, P., Karpinski, M.: 8/7-approximation algorithm for (1, 2)-TSP. In: Proceedings of SODA, pp. 641–648 (2006)
5. Bläser, M.: A 3/4-approximation algorithm for maximum ATSP with weights zero and one. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) RANDOM 2004 and APPROX 2004. LNCS, vol. 3122, pp. 61–71. Springer, Heidelberg (2004)
6. Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document (1976)
7. Erickson, J., Sidiropoulos, A.: A near-optimal approximation algorithm for asymmetric TSP on embedded graphs. In: Proceedings of SOCG, p. 130 (2014)
8. Frieze, A.M., Galbiati, G., Maffioli, F.: On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. Networks **12**(1), 23–39 (1982)
9. Gharan, S.O., Saberi, A.: The asymmetric traveling salesman problem on graphs with bounded genus. In: Proceedings of SODA, pp. 967–975. SIAM (2011)
10. Gharan, S.O., Saberi, A., Singh, M.: A randomized rounding approach to the traveling salesman problem. In: Proceedings of FOCS, pp. 550–559 (2011)
11. Grigni, M., Koutsoupias, E., Papadimitriou, C.H.: An approximation scheme for planar graph TSP. In: Proceedings of FOCS, pp. 640–645 (1995)
12. Karpinski, M., Lampis, M., Schmied, R.: New inapproximability bounds for TSP. J. Comput. Syst. Sci. **81**(8), 1665–1677 (2015)
13. Mömke, T., Svensson, O.: Approximating graphic TSP by matchings. In: 2011 Proceedings of FOCS, pp. 560–569 (2011)
14. Mucha, M.: 13/9-approximation for graphic TSP. In: Proceedings of STACS, pp. 30–41 (2012)
15. Papadimitriou, C.H., Yannakakis, M.: The traveling salesman problem with distances one and two. Math. Oper. Res. **18**(1), 1–11 (1993)
16. Sebő, A., Vygen, J.: 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)
17. Svensson, O.: Approximating ATSP by relaxing connectivity. In: Proceedings of FOCS (2015)
18. Williamson, D.P., Shmoys, D.B.: The Design of Approximation Algorithms. Cambridge University Press, New York (2011)