Approximations to the Metric $s-t$ TSP

Rick Huang, Steven Homberg, Eshaan Nichani

December 2017

1 Introduction

The Traveling Salesman Problem (TSP) describes a set of n cities, and given distances between cities, the goal is to construct the shortest route beginning and ending at an origin city traveling through every other city exactly once. Specifically, given a set of vertices V and edges E with a cost function c describing nonnegative edge lengths in $\mathbb{Q}$, we seek to find a set of edges E_{OPT} that is a *tour*, or an ordering which starts and ends on some $v \in V$ and also traverses all vertices in V, so that $\sum_{e \in E_{OPT}} c(e)$ is minimized. TSP represents an important problem in computational theory, being NP-hard to both solve and approximate, yet includes many applicable variants.

An additional constraint that is often satisfied in practice is requiring the cost function to be metric, implying that the triangle inequality holds for the costs between any set of three vertices in V. Therefore, in the *metric TSP*, given any pair of vertices a and b and any $c \in V$ with pairwise direct costs c_{ab}, c_{ac}, and c_{bc}, we have that $c_{ab} \leq c_{ac} + c_{bc}$. Therefore, we can relax the constraint that each vertex is visited exactly once– the triangle inequality implies that we can modify a tour to not repeat vertices without increasing the cost. Christofides [1] provided a $\frac{3}{2}$-approximation for the problem by first building a minimum cost spanning tree (MST) and then correcting the degree of all vertices to be even, allowing for a valid tour.

The metric TSP can also be generalized to the metric $s-t$ path TSP, which, instead of beginning and ending at the same vertex, begins at $s \in V$ and ends at $t \in V$. If $s = t$, we simply have the metric TSP. Christofides algorithm can be extended to provide a $\frac{5}{3}$-approximation [2], but more recently, Sebo et al [3] discovered a strongly polynomial approximation algorithm achieving a ratio of $\frac{3}{2} + \frac{1}{34}$.

An additional interesting variant of the metric $s-t$ path TSP is the *graphic $s-t$ path TSP*. Consider a graph G with vertices V, edges E, and unit costs on each of the edges. Define the *metric completion* of G as G' with vertices V, edges E, and metric cost function c', where for any two vertices u and v, c'_{uv} is the shortest distance between u and v originally in G. The metric $s-t$ path TSP on G' is then the graphic $s-t$ path TSP, and Gao [4] designed a $\frac{3}{2}$ approximation algorithm for this problem.

In this paper, we focus on outlining the major intuitions and ideas and their prevalence in various approximation algorithms to multiple variants of metric TSP. Specifically, in section 2, we provide some intuition for how a valid $s-t$ tour is constructed from several components. In section 3, we provide a simple application of this approach in Christofides' work on the original metric TSP problem. In section 4, we introduce relevant linear programs (LPs) and other concepts used in modern approaches to construct $s-t$ tours. In section 5, we apply these ideas to provide a more intuitive explanation of Gao's findings for the graphic $s-t$ path TSP. In section 6, we broaden our discussion to the more general metric $s-t$ path TSP, and finally, in sections 7 and 8, we synthesize the main takeaways pertaining to the $\frac{1}{2} + \frac{1}{34}$-approximation algorithm to the metric $s-t$ path TSP found by Sebo et al, and how it builds off of Gao's ideas.

2 Algorithmically building an $s-t$ tour

While the TSP itself isn't obviously tractable, there are related problems which can be efficiently solved which we can use to inform an approximation. In particular, TSP is similar to the problem of finding the minimum cost spanning tree in a graph, as both problems are related to finding an optimal structure which "connects" all the nodes of a graph. However, simply taking the edges of a spanning tree is not enough to construct a tour. It does, though, act as a good starting point.

Given a multisubset U of E (we allow for edges to be repeated in U), we wish to find whether there exists a valid $s-t$ tour using precisely the edges of U. One necessary condition is that the degree of all vertices $v \in V \setminus \{s,t\}$ in the subgraph induced by U must by even, and the degrees of s and t must be odd. However, we can see that this condition is sufficient – any multiset of edges U satisfying this property has a tour using all of its edges in a classic result by Euler, and it can be easily found in a greedy fashion. We use this idea in constructing our approximations, as intuitively we can start with a set of edges which touches all the vertices and then add some to meet this condition and give a tour.

3 Christofides' Algorithm

Christofides' original work on the metric TSP is the simplest application of this general approach to constructing minimum cost tours [1]. In Christofides' algorithm, a minimum spanning tree is first built on the graph. The MST is then extended through degree "parity correction" to form a valid tour. Specifically, this requires considering the set of vertices with odd degrees, or "incorrect parity," and correcting the parity of their degrees to be even by adding additional edges.

Definition 1 *For some subset of edges S, we can define T_S to be the set of wrong degree vertices with respect to these edges, i.e. odd degree vertices which are not s or t, as well as s and t if they are of even degree.*

When S is the spanning tree, then T_S is the set of vertices whose degrees must be corrected in order to be able to construct a tour on those edges.

Definition 2 *We define a T-join for some set of vertices T to be some set of edges J such that the number of edges in J incident on each vertex $i \in T$ is odd, and the number of edges in J incident on each vertex $i \notin T$ is even.*

A T-join captures the notion of the set of edges we must add in order to fix the degrees of the wrong degree vertices - given a spanning tree S, adding a T_S-join produces a valid tour. Thus, Christofides algorithm is as follows:

Algorithm:

1. Compute a minimum spanning tree S of the graph.
2. Compute a minimum perfect matching M within the subgraph with vertices T_S. Note that this is possible, because the parity of $|T_S|$ is necessarily even.
3. Append the edges in M to S, and build a cycle traversing all edges exactly once returning to the starting vertex.

4. Skip repeating vertices, which is possible due to the metric distance condition.

Note that the minimum perfect matching M represents a T_S-join, and we can bound the cost of this by $c(OPT)/2$, where $c(OPT)$ is the optimal TSP cost. The minimum spanning tree is similarly bounded by $c(OPT)$, so this algorithm gives a 3/2 approximation.

While the details of these bounds on the components of Christofides' solution are discrete in nature, these methods of bounding aren't as useful in most future work on the $s-t$ path variant despite the similar high-level structure of the algorithms. The bounds are instead replaced by comparing to an LP-relaxation of the problem.

4 LP-based approaches

4.1 Subtour Elimination LP

An essential component in most of the subsequent work pertaining to the metric TSP and its variants is an LP relaxation which captures some of the essential requirements of the problem known as the *subtour elimination LP*. Introduced by Dantzig, Fulkerson and Johnson [5], this linear program finds a fractional weighting of edges such that each vertex is 'visited' (at least a total weight of 2 is assigned to edges adjacent to each vertex) and every subset of vertices is both entered and exited (at least a total weight of 2 is assigned to edges incident on exactly one vertex in the subset). An integral solution to this LP describes a loop which visits every vertex by the first set of constraints, and which is a single loop by the second set of constraints.

Formally, for a subset of the vertices $U \subset V$, define $\delta(U)$ to be the set of edges incident on exactly one vertex in V, or $\delta(U) = \{\{i,j\} \in E : i \in U, j \notin U\}$. For a graph (V,E) with edge costs $c(e)$, we can define the subtour elimination LP as follows:

$$\begin{aligned}
\min \quad & \sum_{e \in \binom{V}{2}} c(e)x(e) \\
\text{s.t.} \quad & \sum_{e \in \delta(\{i\})} x(e) = f(\{i\}), \text{ for all } i \in V, && (1) \\
& \sum_{e \in \delta(U)} x(e) \geq f(U), \text{ for all } \emptyset \subsetneq U \subsetneq V, && (2) \\
& x(e) \geq 0, \text{ for all } e \in \binom{V}{2}, && (3)
\end{aligned}$$

where $x \in \mathbb{R}^E$ has a component for each edge and $f(U) = 2$ for all subsets U. To instead describe the $s-t$ TSP variant, we define $f(U) = 1$ if $|U \cap \{s,t\}| = 1$ and $f(U) = 2$ otherwise, which ensures that we start/end at s and t and we only require that we enter or leave a subset containing one of them, rather than doing both to ensure a loop.

The following is some useful notation when working with bounds on solutions to this LP:

Definition 3 *Let z be a vector in $\mathbb{R}^E$. Then, for some subset $H \subset E$ we define $z(H) := \sum_{e \in H} z(e)$. Furthermore, given our cost function c, we define $c(z) := \sum_{e \in E} c(e)z(e)$.*

Consider an optimal solution x^* to the subtour elimination LP. The optimal $s-t$ tour on the graph is feasible in the polyhedron, so x^* is a lower bound on the optimal $s-t$ tour. Thus, if we can construct spanning trees and their corresponding T_S-joins such that they are bounded above by some constant multiple of $c(x^*)$, then we must have an approximation to within that same constant factor, which leads us to define the corresponding T-join LP.

4.2 T-join LP

We can similarly relax the discrete notion of a T-join as an LP with some similar structure to the subtour elimination LP, allowing us to similarly derive shared bounds. The relaxation again dictates a fractional weighting of the edges $y(e)$, allowing us to describe the *T-join polyhedron* as

$$\sum_{e \in \delta(U)} y(e) \geq 1 \text{ for all } U \text{ such that } |U \cap T| \text{ is odd},$$
$$y(e) \geq 0 \text{ for all } e \in E.$$

In fact, this optimal solution can be directly useful in constructing these structures. For instance, in the traditional TSP, note that x^* is in the polytope for the LP relaxation of the spanning tree problem, and $x^*/2$ is in the T-join polytope for any even size T (and thus for the set of wrong degree vertices in the spanning tree T_S). This again implies a 3/2 approximation, as both of those LPs are such that their optimal value in the relaxation is in fact achievable with a corresponding discrete structure.

Note, that the same idea does not immediately give us a solution in the $s-t$ path variant. Unlike the loop case, there are subsets U such that $\sum_{e \in \delta(U)} x(e) = x^*(\delta(U)) < 2$, so we no longer have that $x^*(\delta(U))/2 \geq 1$ to satisfy the constraint in the T_S-join polytope. To bound these cases, we introduce the notion of a narrow cut.

4.3 Narrow Cuts

A cut is defined like in max-flow as a subset of the vertices U which contains the start vertex s.

Definition 4 *Given our optimal LP solution x^*, a **narrow cut** is a cut U for which $x^*(\delta(U)) < 2$. We let $\delta(U)$ be the set of all narrow cuts.*

Clearly, a narrow cut will not satisfy the constraints in the T-join polyhedron. However, not all of the cuts are narrow, and not every cut is in fact a constraint in the T-join polyhedron. This observation is the fundamental motivation for much of the LP-based work on approximating $s-t$ path TSP. These algorithms proceed similarly to the discrete ones, where we find a spanning tree and then fix the degrees of the vertices with a T-join. However, in order to maintain a tight bound on the size of the T-join, the spanning tree we choose is constrained to both be small and also prevent narrow cuts from removing the bound on the T-join size.

We first present Gao's application of this principle in the 'graphic' variant of the $s-t$ TSP [4]. We then present Sebo and van Zuylen's approximation for the general metric $s-t$ TSP, which builds on this principle to keep the T-join small while sacrificing connectedness of the spanning tree, and then bounding the cost added later to fix this connectedness [3].

5 Approximating the Graphic $s-t$ Path TSP Variant

Recall that the simple bound on the size of the T-join from noting that $x^*/2$ falls in the corresponding polyhedron no longer applies for the $s-t$ path case because of the narrow cuts (where $x^*(\delta(U))/2 < 1$). However, the T-join polyhedron does not place constraints on every cut, only on the cuts U that have odd intersection with T. In the *graphic* $s-t$ path TSP variant, this enables an elegant way around the issue to give a 3/2 approximation.

The graphic $s-t$ path TSP variant is one where the graph is the metric completion of a unit edge cost graph. The difference here is that our objective function is now just $\sum_{e \in E} x(e)$ instead of $\sum_{e \in E} c(e)x(e)$ in our relaxation of the problem where we allow vertex reuse. Since every spanning tree has the same cost, this enables us to manipulate the spanning tree we choose in order to avoid the narrow cuts ending up as constraints in the T-join polyhedron while still getting a good bound on the spanning tree size. In order to pick a tree, we first examine the structure the narrow cuts induce on the graph.

Lemma 1 *For any two narrow cuts U_1 and U_2, either $U_1 \subseteq U_2$ or $U_2 \subseteq U_1$*

To see why this is true, consider U_1 and $V \setminus U_2$. We have $x^*(\delta(U_1)) + x^*(\delta(V \setminus U_2)) = x^*(\delta(U_1 \cup (V \setminus U_2))) + x^*(\delta(U_1 \cap (V \setminus U_2))) + 2x^*(\{\text{edges between } U_1 \setminus (V \setminus U_2) \text{ and } (V \setminus U_2) \setminus U_1\})$. The first two terms of the RHS must be at least 2 if one is not contained in the other by the LP constraints, while the narrowness of the cuts implies the two terms in the LHS are strictly less than 2, so lemma 1 must hold.

Because the narrow cuts of S are totally ordered, we can group the vertices of the graph by the first cut in which the vertices appear. Let U_i be the ith narrow cut in the ordering. We define $K_i = U_i \setminus U_{i-1}$ for i from 1 to $\ell + 1$, where $U_0 = \emptyset$ and $U_{\ell+1} = V$. Because of the ordering, each K_i is well defined and nonempty.

Let H be the subgraph of G restricted to edges where x^* is nonzero, and let the subgraph of H restricted to vertices in some subset K be $H(K)$.

Lemma 2 *$H(\cup_{p \leq i \leq q} K_i)$ for any $1 \leq p \leq q \leq \ell + 1$ is connected. We call such a set $\cup_{p \leq i \leq q} K_i = U_q \setminus U_{p-1}$ a* level set*.*

This can be verified with casework, but an outline of the argument is as follows. If the subgraph were not connected, we could divide it into two parts W_1 and W_2 with no edges between each other. The subtour elimination LP places a lower bound of 2 on $x^*(\delta(W_1))$ and $x^*(\delta(W_2))$. However, $x^*(\delta(W_1)) + x^*(\delta(W_2))$ is at most $x^*(\delta(U_q)) + x^*(\delta(V \setminus U_p))$ if W_1 and W_2 have no edges between each other, and U_q and U_p are both narrow cuts, which upper bounds their x^* at 2 and thus giving a contradiction.

In particular, note that this holds for $p = q$, so each K_i is a connected component. Similarly, it holds for $p + 1 = q$, which implies that there is always an edge from K_i to K_{i+1}.

The layering structure we described above is sufficient to produce a simple 3/2 approximation in the graphic version of the problem as shown by Gao [4]. In the graphic version, we have that the graph is unit cost as well as metric.

The fundamental idea for this algorithm stems from the connectivity result shown in lemma 2. By building up a spanning tree J from the connected components in the minimal level sets K_i, we can ensure that 'bad' edges do not occur and subsequently give a bound on the cost of the T_J-join that fixes the degree of the vertices.

Algorithm:

1. Solve the subtour elimination LP-relaxation to find an optimal solution x^*
2. Construct the level sets K_i with respect to the subgraph of G induced by the nonzero edges in x^*
3. Take J to be the union of a spanning tree in each L_i, as well as an edge from each K_i to K_{i+1}
4. Find a minimum size T_J join F
5. Return the tour given by the disjoint union of J and F in the doubled graph $2G$

Clearly, solving the LP and finding spanning trees is possible in polynomial time. Additionally, there exists an algorithm to efficiently find the narrow cuts.

The connectivity result in lemma 2 implies that there exists a spanning tree in each K_i, as well as an edge between adjacent K_i. This makes J a spanning tree, so $J \cup F$ includes each vertex as well as having even degree vertices except for s, t. Thus we must arrive at an $s - t$ tour after running the algorithm.

Recall from section 4 that the T_J-join polyhedron satisfies

$$\sum_{e \in \delta(U)} y(e) \geq 1 \text{ for all } U \text{ with } |U \cap T_J| \text{ odd}$$

$$y(e) \geq 0 \text{ for all } e \in E$$

Consider $y = x^*/2$. For any cut U which is not narrow, $x^*(\delta(U)) \geq 2 \Rightarrow x^*(\delta(U))/2 \geq 1$. Otherwise, consider the following:

Lemma 3 (Lemma 2.1 [6]) *For any subset of edges F and subset of vertices U in G, if $|U \cap T_F|$ is odd and U contains exactly one of s, t then $|\delta(U) \cap F|$ is even.*

For any narrow cut U, we have by construction of J that there is exactly 1 edge in $\delta(U)$ which appears in J: the edge going from the K_i corresponding to U to K_{i+1}. Lemma 3 implies that $|\delta(U) \cap J|$ must be even if $|U \cap T_F|$ is odd, so we cannot have that $|U \cap T_F|$ is odd.

Thus, only cuts U which are not narrow have odd intersection with T_J, so we have $\sum_{e \in \delta(U)} x^*/2 \geq 1$ as desired for all such U. $x^*/2$ is then in the polytope and thus gives a T_J-join. The number of edges in this T_J join is at most $\sum_{e \in E} x^*(e)/2$. We also have that the number of edges in J is equal to one less than the number of vertices in G, which is less than $\sum_{e \in E} x^*(e)$. Thus, we have that the total cost of the tour is at most $3/2 \sum_{e \in E} x^*(e)$. By construction of the subtour elimination LP, $x^*(E)$ is a lower bound on the value of the optimal $s - t$ tour, so we have a 3/2 approximation for the graphic case.

6 The general metric s-t path approximation

6.1 Intuition

The key to finding a 3/2 approximation in the graphic case was the fact that narrow cuts did not appear in any of the constraints for our T_S-join LP, due to the fact that $|\delta(U) \cap S| = 1$ for any narrow cut U.

Definition 5 *A cut U is a **lonely cut** of S if $|\delta(U) \cap S| = 1$. We denote the set of lonely cuts with respect to the tree S as $\mathcal{U}(S)$*

Definition 6 *For a spanning tree S, we define the set of **lonely edges** $L(S)$ as edges which separate a narrow cut. In other words, $L(S) = \{e \in E : \exists U \in \mathcal{U}, \delta(U) \cap S = \{e\}\}$.*

However, in the general case it is not always possible to find some spanning tree S that satisfies this property, and the cost of a T_S join may be too high. This suggests that we somehow "merge" the steps ensuring that narrow cuts satisfy some nice additional property and finding a join. To do this, we delete the lonely edges $L(S)$ from our graph, to obtain the forest $F = S \setminus L(S)$. Then, if T_F is the set of wrong degree vertices in F, we find a cheap T_F-join. The hope here is that the many of the edges in our join will actually be lonely edges, which get readded to the graph while simultaneously correcting the parity. Finally, we may encounter a case in which the graph is not connected. To reconnect the graph, we find some spanning tree on the connected components and add the doubled version of this tree to our edge sets in order to also maintain the correct parities. This is the intuition for the Best-of-Many with Deletion (BOMD) Algorithm.

We also saw that, in the discussion on the graphic TSP, that it is possible to choose certain "special" trees that give us nice bounds on the cost of the join. We use a similar strategy here, and want to consider a spanning tree that allows us to more strongly bound the cost of a T_F join and the reconnection cost.

Let's consider when the T_F-join adds back in the lonely edges. Recall the containment property of narrow cuts from lemma 1 - the lonely cuts are a subset of the narrow cuts, and this containment property applies for them as well. Thus after removing $L(S)$, the connected components of F are simply the regions between the lonely cuts.

For lonely cuts U, we remove the edge contained in both $\delta(U)$ and S, and thus $|F \cap \delta(U)| = 0$. Therefore by lemma 3, there is a constraint for U in the T_F join polyhedron. Let J_F be some join - there must be some edge in J_F for each lonely cut. If each edge in J_F crosses at most lonely cut, then the graph $F + J_F$ is actually connected. This is what we want! Unfortunately it's not always the case that edges will have this property.

Definition 7 *Let $B(S)$, the set of **bad edges** is S, be the set of edges crossing more than one lonely cut, i.e. contained in $\delta(U)$ for more than one U.*

Ideally we want to find a J_F join with no bad edges. Unfortunately this can't always be the case. However, we can try to choose S carefully such that it is possible to find a cheap join with minimal bad edges.

It turns out that the condition we want on S for this to occur is for S to be layered.

Definition 8 *A spanning tree S is **layered** if, for any lonely cut U and cut U' with $x^*(\delta(U')) \leq x^*(\delta(U))$, U' is also a lonely cut.*

In other words, any cut which is smaller than a lonely cut must also be a lonely cut itself. Layered trees are important as they prevent bad edges from crossing cuts that are not lonely.

Lemma 4 *Let S be layered, and let $B(S)$ be the set of bad edges of S. Then for any narrow cut U that is not lonely, $x^U(B(S)) = 0$ (define this terminology a little more)*

Lemma 4 becomes instrumental in bounding the cost of reconnection, as we will outline in section 7.

In our proof of the graphic TSP approximation, we found a nice spanning tree using only the vertices of the support of x^*. This motivates trying to find a layered tree on the support of x^* in the general case. In fact, in the next section we discuss something even stronger - writing x^* as a combination of layered trees.

6.2 Finding a Small, Layered Spanning Tree

Recall that we wish to manipulate our choice of spanning tree in such a way to bound on the join and reconnection costs, and that the property we desire is that of being *layered*: a layered spanning tree is one for which the set of lonely cuts $\mathcal{U}(S)$ is downward closed under x^*, i.e. for $U \in \mathcal{U}(S)$, any U' with $x^*(\delta(U')) \leq x^*(\delta(U))$ is also in $\mathcal{U}(S)$.

We claim that x^* can be decomposed into a convex combination of these layered spanning trees, and that this decomposition can be found with an efficient algorithm. Though the complete proof and algorithm of the claim is well beyond the scope of this paper, we provide a quick intuitive sketch of the decomposition. We can extend the notion of level sets introduced by Gao for the graphic case, and consider the level sets induced only by narrow cuts of at most a certain size. We iteratively build a convex combination whose lonely cuts are exactly the narrow cuts of at most a certain size, which clearly then is downward closed under ordering by size.

Therefore, for the remainder of the discussion, we assume our claim holds. Given that we can decompose x^* into a convex combination of layered spanning trees, there must be one such tree which has total cost at most that of x^*.

7 The Best-of-Many with Deletion Algorithm

We can now formalize the general algorithm using the intuition above, titled Best-of-Many with Deletion (BOMD).

Algorithm:

1. Calculate x^*, the optimal solution to the subtour elimination LP above
2. Write x^* as a layered convex combination of $O(|E|)$ spanning trees. We denote by $\mathcal{S}$ the set of spanning trees with positive coefficient.
3. For each $S \in \mathcal{S}$, we calculate the forest based $s-t$ tour outlined in section 6.1:
   (a) Remove the set of lonely edges from S, and denote $F = S \setminus L(S)$
   (b) We calculate y_F, a feasible solution to the T_F-join LP. We write y_F as a convex combination of $O(|E|)$ joins [3].
   (c) Let $\mathcal{J}_F$ be the set of joins with positive coefficient. For each $J_F \in \mathcal{J}_F$:
      i. For each $J_F \in \mathcal{J}_F$, consider the graph with edge set $F+J_F$. We contract each of the connected components, and calculate the minimum cost spanning tree on the contracted components. Let $2D_{F+J_F}$ be the doubled edge set of this MST.
      ii. For each J_F, we obtain the valid solution $(V, F + J_F + 2D_{F+J_F})$. Let $P_1(S)$ be the solution of minimum cost out of all $J_F \in \mathcal{J}_F$
4. For each $S \in \mathcal{S}$, we calculate the Christofides based $s-t$ tour. We find the minimum cost T_S-join, and obtain the valid solution $P_2(S) = (V, S + J_S)$

5. Out of all valid solutions $\cup_{S\in\mathcal{S}}\{P_1(S), P_2(S)\}$, we return the solution with minimum cost.

It is easy to see that this algorithm will produce the desired solution - the graph we construct has the correct parity at each of the vertices due to the addition of J_F, and is connected due to adding $2D_{F+J_F}$. Therefore there exists a tour from s to t, constituting our approximate solution.

8 Analysis of BOMD

We will now present an outline of the proof that the cost of our approximate solution, $c(APX)$ is at most $(\frac{3}{2}+\frac{1}{34})c(x^*)$, where $c(x^*)$ is the cost of the optimal LP solution. If $c(OPT)$ is the optimal $s-t$ tour, then using the fact that $c(APX) \geq c(OPT) \geq c(x^*)$ implies that our approximation is within a factor of $\frac{3}{2}+\frac{1}{34}$ of the optimal solution, and that the integrality ratio is at most this value as well. We will not reproduce the proof in full detail, componentspoints. An interested reader can refer to the paper by Sebo et al.

Recall that we can write x^* as a convex combination of layered spanning trees. In other words, $x^* = \sum_{S\in\mathcal{S}}\lambda_S S$, where $\sum\lambda_S = 1$. In our analysis, we treat $\mathcal{S}$ as a random variable which takes on the value S with probability λ_S. We see that $\mathbb{E}[\mathcal{S}] = c(x^*)$. In our solution, we bound the expected costs of $P_1(\mathcal{S})$, $P_2(\mathcal{S})$, which can be used as upper bounds on the cost of our optimal solution.

We can also use this randomization technique when considering joins. To do this, we construct a solution y_F to the T_F-join LP, and write y_F as a convex combination of valid joins. This allows us to define a random variable $\mathcal{J}_F$ on these valid joins with expectation y_F. Finally, we relate these expected values to our optimal approximation via the probabilistic method - if the expected cost of a solution is C, then there must exist a solution with cost at most C.

The general of the structure of our proof is as follows. We first develop a simple bound of the average cost of a Christofides based tour. We then bound the cost of a join by constructing a solution to the T_F-join LP. Finally, we use the fact that our set of spanning trees $\mathcal{S}$ is layered to obtain a bound on the reconnection costs. We put these together to obtain the average cost of a forest based solution.

8.1 Bounding the cost of a Christofides based tour

First, we bound the expected cost of a Christofides based $s-t$ tour. Let S be a spanning tree, and let $S(s,t)$ be the path from s to t. Note that the edge set $S \setminus S(S,T)$ is a valid T_S join, and thus $c(P_2(S)) \leq c(S)+c(S\setminus S(S,T))$. Recall the $\mathcal{S}$ is the random variable on the spanning trees which comprise x^*. Similarly, we can define the random variable $\mathcal{S}(s,t)$ which corresponds to the path from s to t in the spanning tree randomly chosen by $\mathcal{S}$. Define $p^* = \mathbb{E}[\mathcal{S}(s,t)]$. We then get that

$$\begin{aligned}\mathbb{E}[c(P_2(\mathcal{S}))] &\leq \mathbb{E}[c(\mathcal{S}) + c(\mathcal{S}\setminus\mathcal{S}(s,t))] \\ &= 2\mathbb{E}[c(\mathcal{S})] - \mathbb{E}[c(\mathcal{S}(s,t))] \\ &= 2c(x^*) - c(p^*).\end{aligned}$$

We now aim to bound the forest based tours in terms of x^*, p^*.

8.2 Bounding the average cost of a join

We now must bound the cost of a join. Given a spanning tree S and its corresponding forest F, we aim to construct a solution y_F to the join LP. The cost of y_F will then act as an upper bound on the cost of the

optimal join. This requires satisfying the LP constraints for narrow cuts U. We begin by defining a parity correction vector, given by

$$BP = \frac{1}{2}x^* + \gamma S(s,t),$$

for some parameter γ that we choose later in order to optimize our approximation ratio. If we let $\gamma = 1/2$, then we see that BP is a valid solution to the join LP, as $S(s,t)$ crosses any cut exactly once, and thus $BP(\delta(U)) \geq 1$. This leads to an LP derivation of the $\frac{5}{3}$-approximation given by Hoogeveen [2].

To achieve a better ratio, we pick some $\gamma < \frac{1}{2}$ and add additional components to the vector. The only cuts U for which its corresponding constraint is still violated is one where $x^*(\delta(U)) < 2 - 2\gamma$, and, by Lemma 3, $|\delta(U) \cap F|$ is even. For those cuts, we can define a parity completion vector which is added to BP to ensure the constraints are satisfied.

We split this up into two cases. If $\delta(U) \cap F$ is empty, the, we can define the parity completion for empty cuts, or PCE, as

$$PCE_U = \left(1 - \frac{1}{2}x^*(\delta(U)) - \gamma\right) e_S^U.$$

Since $|\delta(U) \cap F|$ is empty, $\delta(U)$ could only have contained 1 edge in S, a lonely edge, and thus $|\delta(U) \cap S| = 1$. We let e_S^U denote the indicator vector for the lonely edge of S in $\delta(U)$. Then, by adding in PCE_U we see that the constraint is satisfied for all small empty cuts, as since e_S^U only has one non-zero term the expression contributes a net of $1 - \frac{1}{2}x^*(\delta(U)) - \gamma$ to the value of $y_F(\delta(U))$.

In the second case, we define the parity completion vector for non-empty cuts to be

$$PCL_U = \left(\frac{1 - \frac{1}{2}x^*(\delta(U)) - \gamma}{2 - x^*(\delta(U))}\right) x^U,$$

where x^U is defined to be the expectation of e_S^U for all spanning trees in $\mathcal{S}$. We won't delve into the details of why this expression works here, but an intuition is that this is the same expression as PCE_U above, but reweighted to deal with potentially multiple edges of S crossing $\delta(U)$. We add this for all non-empty even cuts.

Therefore we can write

$$y_F = BP + \sum_{x^*(\delta(U)) \leq 2-2\gamma, |\delta(U)\cap F| = 0} PCE_U + \sum_{x^*(\delta(U)) \leq 2-2\gamma, |\delta(U)\cap F| = 2k} PCL_U,$$

and y_F must be a feasible solution to the T_F-join LP. We can now bound the cost of our join by $c(y_F)$.

8.3 Bounding the Reconnection Cost

We next use the layered property of $\mathcal{S}$ to bound the reconnection cost. We revisit the discussion of bad edges from section 6.1. Recall that U is a lonely cut if $\delta(U)$ intersects our spanning tree S in exactly one edge, and that an edge e is bad if it is contained in $\delta(U)$ for more than one U. Let $\mathcal{U}(S,e)$ be the set of lonely cuts of S that contain e. If e is a bad edge, then $|\mathcal{U}(S,e)| \geq 2$.

Let's analyze how reconnection fails for a bad edge. If we choose no bad edges, then the join must have exactly one edge in each lonely cut. This in turn implies that the forest and join together form a single connected component, as the spanning tree from which we created the forest was connected except for the removed edges from lonely cuts. If instead we choose a bad edge b in our join, then we fail to connect the components between each of the cuts in $\mathcal{U}(S,e)$. Because the cuts are totally ordered, the number of components is one more than the number of cuts. Thus to reconnect these components, we need add back in all of the lonely edges in $S \cap \delta(U)$ for $U \in \mathcal{U}(S,e)$, except for maybe one of these edges because the bad

edge serves to connect one pair of the components. We double each of these edges we add to preserve parity - this then becomes an upper bound on the cost of $2D_{F+J_F}$. It is optimal to ignore the edge of maximum cost for each bad edge, but it is actually more convenient in our analysis to pick this at random too.

The reader may refer to section 4 of Sebo's paper for a detailed mathematical analysis, but putting these ideas together, we obtain a bound of

$$\mathbb{E}[c(2D_{F+\mathcal{J}_F})] \leq \sum_{U \in \mathcal{U}} (x^*(\delta(U)) - 1)c(e_S^U),$$

where $\mathcal{U}$ is the set of all narrow cuts and this expectation is taken over the joins $J_F \in \mathcal{J}_F$.

8.4 Putting it together: Bounding the cost of a forest based tour

For each $S \in \mathcal{S}$ we obtain a forest based $s-t$ tour with edges given by $F + J_F + 2D_{F+J_F}$. We can define the random variable $\mathcal{F}$, which simple takes on the value of the forest $F = S \setminus L(S)$ whenever $\mathcal{S}$ is S. By linearity of expectation, the expected cost of our solution (where we choose S at random from $\mathcal{S}$) is $\mathbb{E}[c(\mathcal{F})] + \mathbb{E}[c(J_\mathcal{F})] + \mathbb{E}[c(2D_{\mathcal{F}+\mathcal{J}_\mathcal{F}})]$.

To bound the expected cost of the forest, we use that $\mathcal{F} = \mathcal{S} \setminus L(\mathcal{S})$. The expected cost of $\mathcal{S}$ is $c(x^*)$, and the expected cost of the lonely edges is $\sum_{U \in \mathcal{U}} c(x^U)$. Thus $\mathbb{E}[c(\mathcal{F})] = c(x^*) - \sum_{U \in \mathcal{U}} c(x^U)$

Adding this with our bounds in 8.2 and 8.3 above, we obtain that the expected cost of our solution is at most

$$\frac{3}{2}c(x^*) + \gamma c(p^*) + \sum_{U \in \mathcal{U}} M_U c(x^U),$$

where M_U can be expressed in terms of variables and components of x^*. Sebo demonstrates that if we set $\gamma = 1/16$, each of the multipliers M_U on the $c(x^U)$ is nonpositive and thus the expected cost of $P_1(\mathcal{S})$ is at most $\frac{3}{2}c(x^*) + \frac{1}{16}c(p^*)$.

Earlier we showed that there exists a Christofides based tour of cost at most $2c(x^*) - c(p^*)$. Therefore the optimal solution returned by our BOMD algorithm has cost at most

$$\min\left(\frac{3}{2}c(x^*) + \frac{1}{16}c(p^*), 2c(x^*) - c(p^*)\right) = \left(\frac{3}{2} + \frac{1}{34}\right)c(x^*),$$

as desired.

9 Conclusion

In this paper, we have broken down the main intuitions behind the ideas used in the papers by Gao and Sebo et al. We began by providing a quick demonstration with Christofides' algorithm. We then discussed the subtour elimination LP, and analyzed how narrow cuts tie into ideas concerning the $\frac{3}{2}$-approximation algorithm on the graphic $s-t$ path TSP described by Gao [4]. We finally combined these ideas to describe the Best-of-Many with Deletion algorithm which gives an approximation ratio of $\frac{3}{2} + \frac{1}{34}$ of the metric $s-t$ TSP given by Sebo et al.

We conclude that in metric TSP, we can take advantage of parity correction on a minimum cost spanning tree to build sufficient solutions, and all the secondary tools we have described, including narrow cuts and T-joins help fix and bound the overall approximations.

9.1 Acknowledgments

We would like to thank Professor David Karger and teaching assistants Justin Kopinsky and Luke Schaeffer for their guidance and support throughout the entire semester.

References

[1] N. Christofides. Worst case analysis of a new heuristic for the traveling salesman problem. Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.

[2] J. A. Hoogeveen. Analysis of Christofides' heuristic: Some paths are more difficult than cycles. *Operations Research Letters*, 10(5):291 – 295, 1991.

[3] A. Sebo and A. van Zuylen. The salesman's improved paths: 3/2 and 1/34 integrality gap and approximation ratio. Grenoble Alpes, Optimisation Combinatoire. Univ. Grenoble Alpes, April 2016

[4] Z. Gao. An LP-based 3/2-approximation algorithm for the $s-t$ path graph Traveling Salesman Problem. *Oper. Res. Lett.*, 41(6):615-617, 2013.

[5] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393–410, 1954.

[6] J. Cheriyan, Z. Friggstad, and Z. Gao. Approximating minimum-cost connected T-joins. In *APPROX-RANDOM*, pages 110–121, 2012.