Graph-TSP Review

Michael Kosmider

February 2021

Introduction

This paper is a partial review of Mönke and Svensson's recently published (2011) paper *Approximating Graph TSP by Matchings*. In their paper, they present several new results on the restricted version of the Traveling Salesman Problem (TSP) called graph-TSP. See the abstract for a complete list of results. The purpose of this paper is to present some background material and rewrite certain ideas in a more detailed manner. It is meant to be read in tandem with Mönke and Svensson's paper and to serve as a (hopefully useful) prologue.

Notation

Given a weighted graph $G = (V, E)$, the weight of edge $e \in E$ is denoted by c_e. For any subset $S \in V$, the set of edges in the cut $(S, \bar{S})$ is denoted by $\delta(S)$. We will use $\delta(v)$ instead of $\delta(\{v\})$ for vertices. The degree of a vertex is denoted by $d(v)$. For directed graphs, $\delta^+(v)$ and $\delta^-(v)$ denote the sets of edges entering and leaving the vertex v, respectively.

Problem Definition

There are several equivalent ways of defining graph-TSP. One way is to restrict inputs to complete and weighted graphs $G = (V, E)$ such that the weights of the edges are given by a graphic metric. That is, G is a valid input if and only if there exists an unweighted graph $H = (V, F)$ on the same vertex set such that for any edge $e = \{u, v\}$ in G, c_e is the number of edges in the shortest path between u and v in H. The goal is then of course to find the minimum cost tour for such graphs. An example of a graph G with an underlying graph H is shown in Figure 1.

Notice that any tour C found in G can be translated into a valid (spanning and closed) walk in H of equal cost (where the cost is now the number of edges on the walk, as H is unweighted). Indeed, if C uses an edge $\{u, v\} \notin F$, the walk simply takes the shortest path between u and v, splitting ties arbitrarily. Figure 2 shows a tour in G translated into a walk in H. The reverse is also true for any valid walk W in H that does not redundantly visit vertices. That is, when W visits a vertex, it is either visiting it for the first time or it is revisiting it. If the paths taken by W between newly visited vertices are shortest paths, then W can be translated into a tour of equal cost in G by taking the edges in G that correspond to these shortest paths. Thus, alternatively, graph-TSP is the problem of finding the minimum cost valid walk in an unweighted graph. But from this we can obtain yet another way of thinking about it. Given a solution W in H, we can create a spanning Eulerian multigraph in H by duplicating edges in W the number of times they were used. The goal is thus to find a spanning Eulerian multigraph in H with the minimum number of edges (a Eulerian path in such a multigraph corresponds exactly to a walk in H where we can use edges more than once). This is the formulation used in Mönke and Svensson's paper.

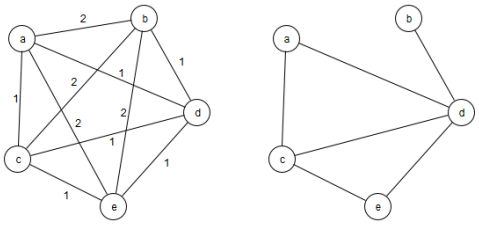

Figure 1: G (left) with an underlying unweighted graph H (right).

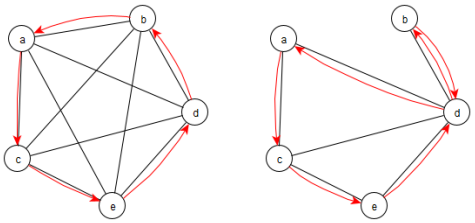

Figure 2: A tour in G translated into a walk in H (both of which start at vertex a). Notice that there is no edge between a and b in H, and so the walk takes the shortest path between them.

A Lower Bound

In order to lower bound the cost of a graph-TSP solution, Mömke and Svensson make use of a well known linear program called the Held-Karp relaxation. We will start by looking at the integer program and how it solves the more general TSP. Given a weighted graph $G = (V, E)$, we can formulate the integer program as follows. For each edge $e \in E$, there is a variable x_e, and the objective function that we wish to minimize is

$$\sum_{e \in E} c_e x_e,$$

which is subject to the constraints

$$x(\delta(v)) = 2 \quad \forall v \in V, \tag{1}$$
$$x(\delta(S)) \geq 2 \quad \forall S \subseteq V \text{ such that } S \neq \emptyset \text{ and } S \neq V, \text{ and} \tag{2}$$
$$x_e \in \{0, 1\} \quad \forall e \in E, \tag{3}$$

where $x(F)$ is shorthand for $\sum_{e \in F} x_e$ for any subset $F \subseteq E$. Observe that a valid tour C in G corresponds to a solution vector x to the integer program, where $x_e = 1$ if the edge e is included C, and $x_e = 0$ otherwise. But why is x feasible? Since C is a cycle, then exactly one edge enters and leaves every vertex. This shows that x satisfies (1). Now we will consider a simple argument that shows why x satisfies (2). Let $S \subseteq V$ be proper and non-empty, and let $v \in \bar{S}$ be arbitrary. If we traverse C by starting and ending with v, we will have to enter and leave S at least once. That means that at least two edges in $\delta(S)$ have corresponding integer program variables of value 1.

On the other hand, any feasible solution vector x corresponds to a valid tour $C = \{e : x_e = 1\}$. Indeed, (1) guarantees that the edges in C form a set of disjoint cycles. This is because (1) forces C to contain two

edges incident to each vertex. Now assume, by contradiction, that one of these cycles, C', does not visit every vertex in V. Let $S' \subseteq V$ be the set of vertices visited by C'. Then $x(\delta(S')) = 0$, contradicting (2). Therefore, C is itself a cycle that visits every vertex. This shows that the integer program and TSP are equivalent.

In the linear program relaxation, we change (3) to

$$x_e \geq 0 \quad \forall e \in E,$$

which gives us a lower bound on the optimal tour. Mönke and Svensson use a modified version of this linear program, adapted to graph-TSP inputs specifically. They cite a result from Goemans and Bertsimas[2] which shows that we can drop constraint (1) entirely for metric graphs, and the linear program will have the same optimal value as before. Furthermore, on an input G = (V, E) (complete and weighted) that has an underlying graph $H = (V, F)$ (unweighted, possibly incomplete), they show that any feasible solution vector x can be transformed into a solution vector x' of equal cost where $x'_{\{u,v\}} = 0$ for any edge $\{u, v\} \notin F$. This can be done as follows. Begin by letting $x' = x$. Then, pick a variable $x'_{\{u,v\}}$ such that $x'_{\{u,v\}} > 0$ and $\{u, v\} \notin F$, and set $x'_{\{u,v\}} = 0$. Finally, add $x_{\{u,v\}}$ to x'_e for each edge e on the shortest path in H between u and v. If there remains a non-zero variable whose corresponding edge is not in F, then we can repeat this process on x' to get x'' and so on. Assume that x' is the final solution obtained by this process (i.e., it finishes in one iteration). Now two questions should come to mind: why is the cost of x' unchanged, and why does x' still satisfy (2)? To answer the first question, by setting $x'_{\{u,v\}} = 0$, the value of the objective function decreases by $c_{\{u,v\}} * x_{\{u,v\}}$. But there are exactly $c_{\{u,v\}}$ edges on the shortest path in H between u and v and thus $x_{\{u,v\}}$ is re-added $c_{\{u,v\}}$ times. To answer the second question, consider any proper and non-empty subset $S \subseteq V$. Since x is feasible, we have that $x(\delta(S)) \geq 2$. If $\{u, v\} \notin \delta(S)$, then the constraint must remain satisfied for x' since no other variables decrease in value. On the other hand, if $\{u, v\} \in \delta(S)$, where WLOG $u \in S$, we may worry that the constraint is no longer satisfied because $x'_{\{u,v\}} = 0$. But consider the shortest path between u and v in H. There must be at least one edge e on this path that leaves S, because $u \in S$ and $v \notin S$. In obtaining x', the value of x'_e becomes $x_e + x_{\{u,v\}}$, so the net value of $x'(\delta(S))$ cannot be smaller than that of $x(\delta(S))$. Since the cost and feasibility are unchanged after one iteration of the process, then the same holds inductively for any number of iterations. This brings us to the final form of the linear program: minimize

$$\sum_{e \in F} x_e,$$

subject to the constraints:

$$x(\delta(S)) \geq 2 \quad \forall S \subseteq V \text{ such that } S \neq \emptyset \text{ and } S \neq V, \text{ and} \tag{4}$$

$$x_e \geq 0 \quad \forall e \in F, \tag{5}$$

keeping in mind that $c_e = 1$ for all edges in F. Thus, in place of complete and weighted graphs, we just consider unweighted graphs $G = (V, E)$ knowing that the linear program has the same optimal value on the complete and weighted metric completion of G. Mönke and Svensson refer to this linear program as $LP(G)$, and to its optimum as $OPT_{LP}(G)$.

Matchings

Matchings play a central role in TSP. Given an unweighted graph $G = (V, E)$, a matching is a subset of E such that no two edges in the subset are incident to the same vertex. Mönke and Svensson make use of matchings in order to obtain a spanning Eulerian multigraph in G with a guaranteed upper bound on the number of edges.

Here we will look at a Lemma 2.2, which is specific to cubic 2-edge-connected graphs (i.e., graphs whose vertices all have degree three and remain connected after the removal of any edge). This lemma builds up nicely from several different results. As cited by Mönke and Svensson, a result from Edmonds[3] shows that given any graph $G = (V, E)$, the set of inequalities (on the variables $\{x_e : e \in E\}$)

$$x(\delta(v)) = 1 \text{ for } v \in V, \quad x(\delta(S)) \geq 1 \text{ for } S \subseteq V \text{ with } |S| \text{ odd}, \quad \text{and } x \geq 0$$

determines the perfect matching polytope. This means that every extreme point of the polytope is a vector x such that all values in x are either 0 or 1, and that the set of edges $\{e : x_e = 1\}$ is a perfect matching in G. Furthermore, every perfect matching in G corresponds to an extreme point on the polytope. Next, a result from Naddef and Pulleyblank[4] shows that for cubic 2-edge-connected graphs, the vector x such that $x_e = \frac{1}{3}$ for all elements satisfies the inequalities. The final piece of the puzzle comes from an algorithmic version of Carathéodory's theorem for convex hulls. The theorem states that there exists a polynomial time algorithm such that given a feasible solution x to a polytope defined by such a set of inequalities, we can decompose x into a convex combination of extreme points of the polytope. That is, we can find extreme points $x_1, x_2, \ldots, x_k$ and real numbers $a_1, a_2, \ldots, a_k$ such that $a_1 + a_2 + \cdots + a_k = 1$ and $a_1x_1 + a_2x_2 + \cdots + a_kx_k = x$, where x was the solution where $x_e = \frac{1}{3}$.

Now suppose we fix some edge e in G and randomly pick an extreme point such that the probability of picking x_i is a_i. What is the probability that e is in the matching defined by x_i? In order for this to occur, we must pick an extreme point from the set $S = \{x_i : x_{i_e} = 1\}$. But the probability of picking an extreme point from S is precisely $\sum_{i:x_i \in S} a_i = \frac{1}{3}$ since $x_e = \frac{1}{3}$. This finally proves Lemma 2.2, which states that given a cubic 2-edge-connected graph G, we can, in polynomial time, find a distribution of perfect matchings such that if we fix an edge and randomly pick a matching from that distribution, the edge has a $\frac{1}{3}$ probability of being in that matching. This lemma plays a crucial role in the development of the next result.

Finding a Spanning Eulerian Multigraph

We will now analyze another central result: Theorem 3.2. We will see how this theorem follows from Lemma 3.3 (which is really Lemma 2.2 in disguise). Assume that we have a 2-vertex-connected graph $G = (V, E)$ and a removable pairing (R, P). A 2-vertex-connected graph is a connected graph that does not contain a cut vertex, and a cut vertex is a vertex whose removal results in the creation of at least one additional connected component. See Definition 3.1 for a definition of removable pairings. The goal is now to obtain a spanning Eulerian multigraph in G whose number of edges is bounded by $\frac{4}{3}|E| - \frac{2}{3}|R|$. In order to do so, we will need to use the distribution of matchings given by Lemma 2.2, which requires that G is cubic and 2-edge-connected. The latter holds for G since 2-vertex-connected graphs are 2-edge-connected except when G consists of a single edge (which we will assume is not the case). However, in order to use Lemma 2.2, we turn G into a cubic graph $G' = (V', E')$ as follows:

- If v is a vertex of degree 3, it remains untouched.
- If v is a vertex of degree 2 with neighbors u and w, we replace it by a gadget as follows:

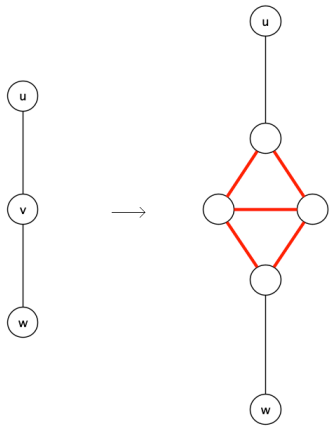

Figure 3: Replacing v with a gadget. Edges originally in G are colored in black.

- If v is a vertex of even degree $d > 3$, we replace v with a tree that consists of a cubic root r and $\frac{d}{2}$ leaves, where every internal vertex has degree 3. The exception is when $d(v) = 4$, in which case the tree has a binary root r, and we follow up by replacing r with the gadget used for degree 2 vertices. Now what happens to edges that were incident to v? For each pair $p \in P$ that was incident to v, we attach it to a unique leaf. Each unused leaf thereafter receives 2 of the remaining edges in any order. Below is an example with $d = 8$.

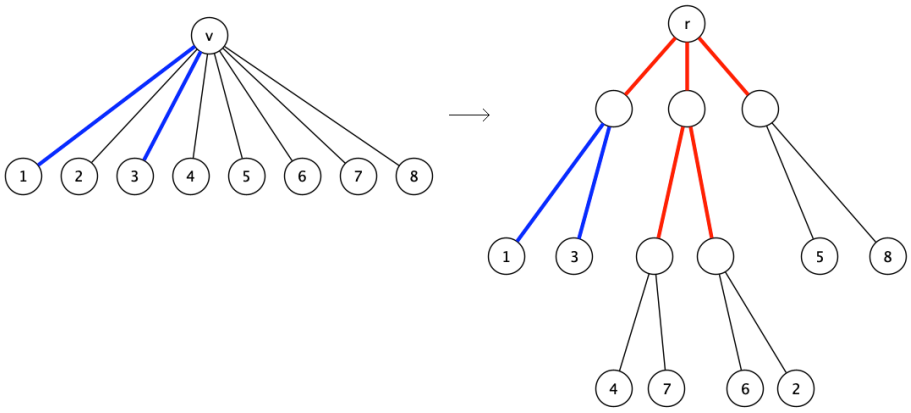

Figure 4: Replacing v with a tree. An example pair $p \in P$ is highlighted in blue. Tree edges are drawn in red. It is worth noting that the tree drawn in red can be created in a systematic way: start with a root that has 3 children. If we need more leaves, we can pick any leaf and add two children to it thereby creating an internal node of degree 3.

- Finally, if v is a vertex of odd degree $d > 3$, we do something similar as in the previous case. We replace v with a tree that has a binary root r, degree 3 internal vertices and $\frac{d-1}{2}$ leaves. As before, any pair $p \in P$ gets attached to its own leaf, and each unused leaf thereafter receives 2 remaining edges in any order. This time, there will be one edge remaining which gets attached to the root, thus ensuring that r itself has degree 3. Below is an example with $d = 9$

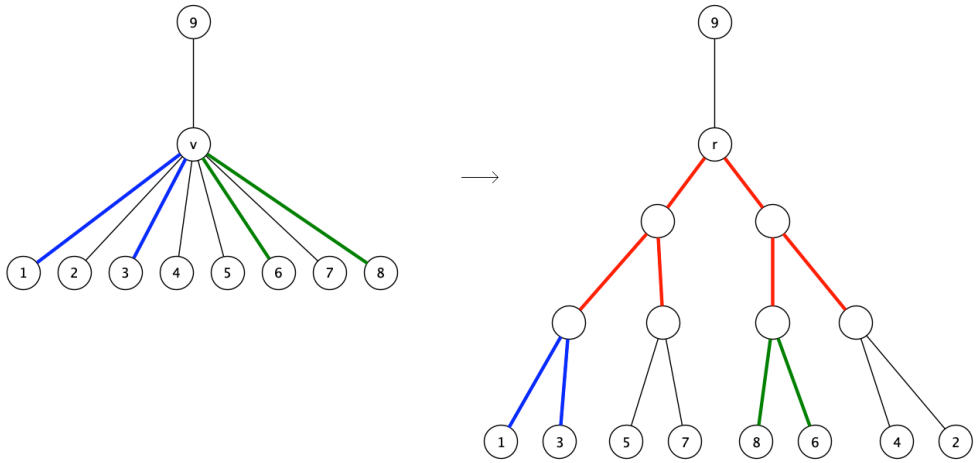

Figure 5: Replacing v with a gadget. There are two pairs in P, one highlighted in blue and the other in green.

Now that G' is cubic, we can invoke Lemma 2.2. For convenience and peace of mind, let us consider the situation in a more formal probabilistic setting. Given G', Lemma 2.2 gives us a sample space S with a probability function P, and matchings $M_1, M_2, ..., M_k \in \mathcal{M}$ such that:

1. $A_i \in S$ is the outcome of picking M_i for each $M_i \in \mathcal{M}$.
2. If $e \in E'$ and $X_e : S \to \mathbb{N}$ is a random variable defined

$$X_e(A_i) = \begin{cases} 0 \text{ if } e \notin M_i \\ 1 \text{ if } e \in M_i \end{cases},$$

then $P(X_e = 1) = \frac{1}{3}$

Now consider what happens when we modify each matching in $\mathcal{M}$ to its restriction in G. That is, for each matching $M_i \in \mathcal{M}$, change M_i to $M_i \cap E$. Fix an edge $e \in G$, and randomly sample a matching from $\mathcal{M}$. Note that the set $\{A_i : e \in M_i\}$ does not actually change since we did not remove any edges in G from the matchings, and thus $P(X_e = 1) = P(\{A_i : e \in M_i\})$ does not change either! Thus $P(X_e) = \frac{1}{3}$ still holds for edges in G. We now have have a distribution of sets of edges in G such that each edge in G has a $\frac{1}{3}$ probability of being in a set randomly picked from this distribution. As a quick observation notice that these sets are no longer necessarily perfect matchings, or even matchings at all in G. Indeed, if G' contains a gadget as shown in Figure 3, then the edges $\{u, v\}$ and $\{w, v\}$ can both be in the same matching in G', but they are incident to the same vertex in G. However, this is not an issue and what matters is that the probability statement holds. Keeping in mind that $\mathcal{M}$ was restricted to G, we can deduce two additional and very useful facts about the sets in $\mathcal{M}$. Let $M_i \in \mathcal{M}$ be any set. Then:

1. At most one edge from each pair in P is in the set M_i. This is because when we constructed G', we made sure that for each pair $p \in P$, the two edges in p were incident to the same vertex. Therefore, as M_i is a matching in G', it cannot contain both edges.
2. Every vertex has degree 2 in the multigraph G_m formed by adding M_i to the edges of G. The key to this is that before being modified, M_i was a perfect matching in G'. As every vertex in G' has degree 3, then adding M_i to the edges of G' would have created a multigraph G'_m in which every vertex has degree 4. But G_m is G'_m with the gadgets compressed back into a single vertex. It thus remains to show that when compressing the gadgets, the degrees of all vertices remain even. Indeed, if you take any pair u, v of even degree vertices that are connected by an edge (or multiple edges, since we are

looking at multigraphs), then compressing them into one vertex must result in a vertex that is also of even degree. This is because any edge between the two vertices contributes twice to their combined degree. I.e., if u has degree $2k_1$ and v has degree $2k_2$, and there are l edges between them, then the resulting vertex has degree $2k_1 + 2k_2 - 2l$. This fact is useful because a gadget can be compressed by systematically compressing pairs in the gadget until it one vertex remains, and we showed that the degree of each vertex remains even throughout the whole process.

Now before proceeding, let us summarize what we have so far. We started with a 2-vertex-connected graph G and a removable pairing (R, P). We then used G to obtain a cubic 2-edge-connected graph G', on which we invoked Lemma 2.2. This Lemma in turn produced, in polynomial time, a useful distribution of perfect matchings $\mathcal{M}$ in G'. Finally, we modified $\mathcal{M}$ by restricting it to G and got a distribution $\mathcal{M}$ of subsets of edges in G such that:

1. If we fix an edge in G and randomly pick a subset from the distribution, the probability that the edge is in the distribution is $\frac{1}{3}$.
2. Any subset M picked from $\mathcal{M}$ satisfies two properties:
   (a) At most one edge from each pair in P is in M.
   (b) Every vertex in the multigraph formed by adding M to the edges of G has even degree.

This is precisely what is stated by Lemma 3.3. We now have the tools we need to obtain the aforementioned multigraph in G. We analyze what happens when we randomly pick a subset M from $\mathcal{M}$. Once picked, split M into two disjoint sets: $M_r = M \cap R$ and $\bar{M_r} = M \cap (\bar{R})$. That is, M_r is the subset of edges in M that are also in R, and $\bar{M_r}$ is the subset of edges in M that are not in R. Next, let H be the multigraph defined $H = (V, (E \cup \bar{M_r}) \setminus M_r)$. That is, to obtain H, we take G and add the edges in $\bar{M_r}$ and then remove the edges in M_r. We will now show that the degree of each vertex in H is even and that H is connected, proving that H is a spanning Eulerian multigraph. Then, we will bound the expected number of edges in H, from which it will follow that the desired multigraph can be found.

A clever argument shows that the degree of every vertex in H is even. Indeed, fix some vertex v in G, and assume $F \subseteq M$ is the set of edges of M incident to v. Suppose that we add F to G. By (2.b), we know that $d(v)$ becomes even if it was not already. But for each edge in F added to G, the parity of $d(v)$ swaps exactly once, and thus in the process of adding F to G, the parity of $d(v)$ swaps exactly $|F|$ times. But the key observation is that removing an edge incident to v also swaps the parity of $d(v)$! In obtaining H, each edge in F is either added or removed from G, and thus the parity of $d(v)$ also swaps $|F|$ times. This proves that $d(v)$ is even in H as well.

To see that H is connected, consider the edges that were removed. We only removed edges in M_r, and (2.a) states that M (and thus M_r) contains at most one edge from each pair in P. By the definition of removable pairings, H is connected.

Finally, we will show that the expected number of edges in H is $\frac{4}{3}|E| - \frac{2}{3}|R|$. We return to the more formal probability setting where we have a sample space S and an outcome for $A_i \in S$ for each set $M_i \in \mathcal{M}$. In addition to the random variables X_e for each edge in E, we define three random variables $X, Y, Z : S \to \mathbb{R}$ as follows:

$$X(A_i) = |\bar{M}_{i_r}|, \quad Y(A_i) = |M_{i_r}|, \quad Z(A_i) = |E| + X(A_i) - Y(A_i).$$

We are thus interested in $E(Z)$. We first determine $E(X)$ and $E(Y)$. The key is that

$$\begin{aligned}
X(A_i) &= |\bar{M}_{i_r}| \\
&= |\{e \in M_i \cap \bar{R}\}| \\
&= \sum_{e \in M_i \cap \bar{R}} X_e(A_i), \text{ since } X_e(A_i) = 1 \text{ for } e \in M_i, \\
&= \sum_{e \in M_i \cap \bar{R}} X_e(A_i) + \sum_{e \in (E \setminus M_i) \cap \bar{R}} X_e(A_i), \text{ since } X_e(A_i) = 0 \text{ for } e \notin M_i, \\
&= \sum_{e \in E \cap \bar{R}} X_e(A_i).
\end{aligned}$$

This means that $X = \sum_{e \in E \cap \bar{R}} X_e$, which simplifies things greatly as we do not have to determine $E(X)$ directly! We can thus apply linearity of expectation and obtain

$$\begin{aligned}
E(X) &= E\left(\sum_{e \in E \cap \bar{R}} X_e\right) \\
&= \sum_{e \in E \cap \bar{R}} E(X_e) \\
&= \sum_{e \in E \cap \bar{R}} P(X_e = 0) * 0 + P(X_e = 1) * 1 \\
&= \sum_{e \in E \cap \bar{R}} \frac{2}{3} * 0 + \frac{1}{3} * 1 \\
&= \sum_{e \in E \cap \bar{R}} \frac{1}{3} \\
&= |E \cap \bar{R}| * \frac{1}{3} \\
&= \frac{1}{3}(|E| - |R|), \text{ since } R \subseteq E.
\end{aligned}$$

We follow a similar procedure to obtain $E(Y)$:

$$\begin{aligned}
Y(A_i) &= |M_{i_r}| \\
&= |\{e \in M_i \cap R\}| \\
&= \sum_{e \in M_i \cap R} X_e(A_i) \\
&= \sum_{e \in M_i \cap R} X_e(A_i) + \sum_{e \in (E \setminus M_i) \cap R} X_e(A_i) \\
&= \sum_{e \in E \cap R} X_e(A_i).
\end{aligned}$$

Thus $Y = \sum_{e \in R} X_e(A_i)$, since of course $E \cap R = R$. As before,

$$\begin{aligned}
E(Y) &= E\left(\sum_{e \in R} X_e\right) \\
&= \sum_{e \in R} E(X_e) \\
&= \frac{1}{3}|R|.
\end{aligned}$$

Applying linearity of expectation once more, we get:

$$\begin{aligned}
E(Z) &= E(|E| + X - Y) \\
&= |E| + E(X) - E(Y) \\
&= |E| + \frac{1}{3}(|E| - |R|) - \frac{1}{3}|R| \\
&= \frac{4}{3}|E| - \frac{2}{3}|R|.
\end{aligned}$$

Since $E(Z) = \frac{4}{3}|E| - \frac{2}{3}|R|$, then there exists an outcome $A_i \in S$ such that $Z(A_i) \leq \frac{4}{3}|E| - \frac{2}{3}|R|$, since otherwise the expected value would be higher. But

$$\begin{aligned}
Z(A_i) &= |E| + X(A_i) - Y(A_i) \\
&= |E| + |\bar{M_{i_r}}| - |M_{i_r}|,
\end{aligned}$$

from which it follows that M_i is a set such that $|E| + |\bar{M}_{i_r}| - |M_{i_r}| \leq \frac{4}{3}|E| - \frac{2}{3}|R|$. Finally,

$$H_i = (V, (E \cup \bar{M}_{i_r}) \setminus M_{i_r})$$

is a spanning Eulerian multigraph in G with the required number of edges. In order to find H_i (or an even smaller multigraph), we can simply iterate over $\mathcal{M}$ and look at all possible multigraphs H obtained as above. Since $\mathcal{M}$ contains polynomially many sets, this process takes polynomial time. We have thus proven Theorem 3.2.

Circulations

We need two ingredients in order to apply Theorem 3.2: A 2-edge-connected graph G, and a removable pairing (R, P) in G. In this section we briefly discuss circulations, which Mönke and Svensson use to obtain a removable pairing.

We first refresh on the definition of circulations. Given a directed graph $G(V, A)$, assigning an upper bound (capacity) u_a and lower bound (demand) l_a to each arc $a \in A$ turns G into a circulation network. A circulation is then an assignment of flows to arcs $f : A \to R$ that satisfies the following properties:

- $l_a \leq f(a) \leq u_a \quad \forall a \in A$.
- $\sum_{a \in \delta^+(v)} f(a) = \sum_{a \in \delta^-(v)} f(a) \quad \forall v \in V$, i.e., the total flow going into a vertex is the total flow leaving the vertex.

Now given a 2-vertex-connected graph G, we can obtain the circulation network $C(G, T)$. See the first four paragraphs of section 4 in Mönke and Svensson's paper for a nice description of how to obtain $C(G, T)$. The demand on each arc in $\overrightarrow{T}$ is then set to 1 and 0 for all other arcs. The capacity of each arc is ∞. Finally we introduce the cost of the circulation. Traditionally, the cost function of a circulation network assigns a cost c_a to each arc, and the cost of a particular circulation f in the network is $c(f) = \sum_{a \in A} c_a \times f(a)$. However, Mönke and Svensson use a different type of cost function for $C(G, T)$. It is defined as $c(f) = \sum_{v \in \mathcal{I}} \max[f(B(v)) - 1, 0]$. Here, $f(B(v))$ is used as shorthand to denote $\sum_{a \in \delta^+(v)} f(a)$. What this means is that the cost of a circulation f is the total sum of flows going into vertices in $\mathcal{I}$, but with a twist: the first unit of flow going into each vertex is free. As an aside, this circulation network can be adapted to have a more traditional cost function as follows. For each vertex $v \in \mathcal{I}$, we can connect all of the back arcs of v to a new vertex v' and then add a pair of arcs from v' to v, as shown below.

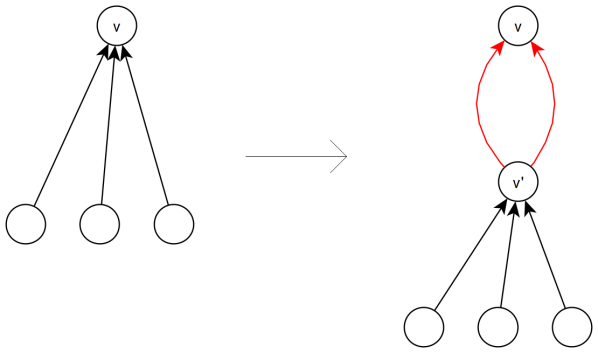

Figure 6: A modification to the network. The two new arcs are shown in red.

Next, we assign a cost of 0 to all of the old arcs. For each new pair of arcs, one of them has cost 1 and capacity ∞, and while the other has cost 0 but capacity 1. The key is that a circulation f in the original circulation network has a corresponding circulation in the modified one (where all but one unit of flow going into any vertex $v \in \mathcal{I}$ is now concentrated on the infinite capacity arc of cost 1), and the costs of both circulations are the same under their respective cost functions.

Conclusion

In this paper we introduced a relatively self contained description of the modified Held-Karp linear program and how it produces a lower bound for graph-TSP. We then discussed matchings and how they can be used to obtain a spanning Eulerian multigraph in 2-vertex-connected graphs. We ended with a brief discussion on circulations. This material was meant to be a detailed walk-through of certain key ideas in the first half of Mönke and Svensson's paper.

References

[1] Tobias Mömke and Ola Svensson. Approximating Graphic TSP by Matchings. 2011. Available at: https://arxiv.org/abs/1104.3090 (Accessed May 3, 2021)

[2] Michel X. Goemans and Dimitris J. Bertsimas. On the parsimonious property of connectivity problems. In *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1990)*, pages 388–396, 1990.

[3] Jack Edmonds. Maximum matching and a polyhedron with 0, 1 vertices. *Journal of Research of the National Bureau of Standards*, 69:125–130, 1965.

[4] Denis Naddef and Wiliam R. Pulleyblank. Matchings in regular graphs. *Discrete Mathematics*, 34(3):283–291, 1981.