A Very Simple LaTeX 2_ε Template

Vitaly Surazhsky
Department of Computer Science
Technion—Israel Institute of Technology
Technion City, Haifa 32000, Israel

Yossi Gil
Department of Computer Science
Technion—Israel Institute of Technology
Technion City, Haifa 32000, Israel

December 13, 2011

Abstract

This is the paper's abstract ...

1 Introduction

The Travelling Salesman Problem (TSP) is perhaps one of the most famous NP-Complete problems. There are many variants of the problem, thus to avoid ambiguity we will make the following definition:

General TSP: Let $G = (V, E)$ be a complete graph on the set of vertices V. Further, assume that there exists a non-zero weight function w that assigns weight to every edge. The TSP problem asks us to find an Eulerian cycle in V of minimal weight that visits each node exactly once.

A particular variant of the general TSP specifies that the weight function w satisfies the triangle inequality, that is: $w(x, y) + w(y, z) \leq w(x, z)$. This is referred to as the metric-TSP.

A lot of work has been done to study metric-TSP. It has been shown that metric-TSP is in the class APX-hard. Basically if a problem is APX-hard,

it means that it is impossible to device a Polynomial Time Approximation Scheme (PTAS) for it unless P=NP. Therefore we cannot arbitrarily approximate metric-TSP. Indeed, it is shown that it is NP-hard to approximate metric-TSP with a ratio better than 220/219.

For more than 30 years, the best known approximation algorithm for metric-TSP is the Christofides algorithm, which achieves an approximation ratio of 3/2.

A linear programming relaxation technique known as the Held-Karp relaxation was instrumental in understanding the bounds on the approximation of metric-TSP. Like many other LP relaxation problems we learned in 6.854, it first solves the LP for continuous values of the variables, then attempts to give only integer solutions which corresponds to an actual choice of a path in the TSP. Thus, it is very important for us to understand the upper and lower bounds of the integrality gap. To this end, the best lower bound is 4/3 and is conjectured to be tight, whereas the best upper bound is obtained via Christofide's algorithm, which is 3/2.

This is where things currently stand for the metric-TSP problem, however, many interesting results have since been found for a particular kind of metric-TSP, and the upper bound have been improved. These special instances of metric-TSP are referred to as graph-TSP, where we have some underlying graph G with vertices V, and the weight function w between two points is equal to the length of the shortest path between these two points. Note that the edges are unweighted, and "shortest" simply means the least *number* of edges.

Many believe graph-TSP captures the "hardness" of metric-TSP since it is APX-hard with a lower bound 4/3 as given by the Held-Karp relaxation. This is to be contrasted with other special cases of metric-TSP such as the Euclidean-TSP (where distances between vertices are the Euclidean distances between them), which is shown to admit a PTAS.

Recently, Gharan et al. showed that graph-TSP can be approximated with a ratio strictly better than 1.5 by ϵ, where ϵ is on the order of 10^{-12}. In this paper by Momke and Svensson, the ratio is further improved to 1.461.

2 Christofides Algorithm

The big idea in all three of these approximation algorithms: Christofides, 3/2-ϵ and 1.461 are somewhat similar, therefore it helps to start building

intuition from the simplest - Christofides.

Christofides algorithm can be used to solve any metric-TSP (not just graph-TSP). Again, let $G = (V, E)$ be a complete graph, and let the weight function be w which satisfies the triangle inequality. Christofides algorithm operates as follows:

1. Compute the minimum spanning tree T of G.

2. Let O denote the set of vertices in T with odd degree. We find a minimum weight perfect matching M in the complete graph over the vertices in O.

3. Combine the edges of T and M to form a multigraph H. (A multigraph is just a graph that can have more than one edges between the same pair of nodes).

4. Since M matches up all the odd degree nodes in T, $H = M \cup T$ now has only even-degree vertices, hence it is an Eulerian circuit.

5. Traverse the circuit found in the previous step and skipping visited nodes via shortcutting to make it Hamiltonian.

We will show that the above algorithm gives a 3/2-approximation.

Let A denote the edge set of the optimal solution of TSP on G. and let $w(S)$ denote the total weight of a set $S \in E$ of edges.

Claim 1: $w(T) \leq w(A)$.

This is true because A spans all the vertices V, and therefore it must contain some spanning tree, which will have a bigger weight than the MST T.

Claim 2: $w(M) \leq w(A)/2$.

This is slightly less obvious. Recall O is the set of vertices in T with an odd degree. Suppose we want to find an optimal solution of TSP for the complete graph over vertices from O only (ie, only need to make sure we visit every node in O), let's call the set of edges in this TSP solution B. Clearly $w(B) \leq w(A)$ because the tour with edge set A visits more nodes, and since the weights satisfy the triangle inequality, you can never save distance by visiting more points.

Now, we show that M, the minimum weight perfect matching over O has weight at most $w(B)/2$ which is in turn less than $w(A)/2$. First, notice that there must be an even number of vertices in O, and since G is complete, a perfect matching exists. In particular, if we list the edges in B in order: e_1,e_2,...,e_{2k}, we get an Eulerian path, and we can easily see that both e_1,e_3,...,e_{2k-1} and e_2,e_4,...,e_{2k} are perfect matchings. Since their combined weight is $w(B)$, one of these perfect matchings will have at most $w(B)/2$.

Hence, M being the minimum weight perfect matching over O will have at most $w(B)/2 \leq w(A)/2$, which proves the claim.

Combining claims 1 and 2, the total weight of the multigraph H is at most $w(T) + w(M) \leq \frac{3}{2}w(A)$. This can only decrease when we take shortcuts in step 5. Thus, this algorithm yields a 3/2-approximation.

3 Held-Karp Relaxation

We can apply LP relaxation and convert the optimal tour of a TSP into an equivalent LP. This is known as the Held-Karp relaxation. The variables are $x_{\{u,v\}}$ corresponding to each edge. Intuitively, $x_e = 1$ if it is in the optimal tour and 0 otherwise. For a general TSP problem, let $G = (V, E)$ be the complete graph on the set of vertices, and let $w_{\{u,v\}}$ be the distance between u and v. Then, the following LP is used to find the optimal tour:

Minimize: $\Sigma_{e \in E} c_e x_e$

subject to:

$x(\delta(v)) = 2$ for $v \in V$,

$x(\delta(S)) \geq 2$ for $\emptyset \neq S \subset V$, and

$x \geq 0$

Where $\delta(S)$ denotes the set of edges crossing the cut $(S, \bar{S})$, and $x(F) = \Sigma_{e \in F} x_e$ for any $F \subseteq E$.

Heuristically, the equality constraint ensures that the cross across every vertex and its complement is 2, meaning that it is visited exactly once. The inequality ensures that there are at least 2 edges connecting every set S and its complement, this implies that S and its complement are connected, thus this ensures the connectedness of the graph, meaning that every node is visited by a single tour.

Notice that if we let $S = v$ in the inequality constraint, we obtain $x(\delta(v)) \geq 2$ which is looser than the equality constraint (≥ 2 instead of $= 2$). However, Goemans and Bertsimas proved that for metric distances, the looser inequality constraints do not change the optimal value of the LP. Also, for graph-TSP, the distances between each pair of vertices in the complete graph is equal to the number of edges in the shortest paths connecting them in the original graph, making $c_e = 1$. Hence, we can formulate the Held-Karp relaxation for graph-TSP as:

Minimize: $\Sigma_{e \in E} x_e$

subject to:

$x(\delta(S)) \geq 2$ for $\emptyset \neq S \subset V$, and
$x \geq 0$

We will refer to this linear program as $LP(G)$, and its optimal solution as $OPT_{LP}(G)$.

We can make an observation that allows us to concentrate the graph-TSP problem for graphs which are 2-vertex-connected. Suppose the graph is not 2-vertex-connected, suppose after removing vertex v the graph will decompose into G_1, G_2, ..., G_l. We can then recursively solve the graph-TSP problem on the subgraphs: $G_1 \cup v$, $G_2 \cup v$, etc. The union of the solutions for the subproblems can then be combined to give a solution for the whole graph. This observation is summarized in the following lemma which we will state without proof:

Lemma 2.1: Suppose G is a connected graph, and G_1,G_2,... are 2-vertex-connected subgraphs of G. Suppose an r-approximate algorithm exists for each subgraph G_i (with respect to $OPT_{LP}(G_i)$), then there is an r-approximate algorithm on G (with respect to $OPT_{LP}(G)$).

4 A Useful Lemma for Matchings

Recall that finding a matching is crucial in Christofide's algorithm, so before proceeding further, we need to mention a result about matchings which will become useful later in our algorithm.

Consider a linear program whose feasible region is suppose to describe perfect matchings. It has variables x_e for every $e \in E$. Intuitively, $x_e = 1$ if e is in the matching, and 0 otherwise. Consider the following constraints:
$x(\delta(v)) = 1$ for $v \in V$,
$x(\delta(S)) \geq 1$ for $S \subseteq V$ with $|S|$ odd, and
$x \geq 0$

To gain some intuition into why the feasible regions of the above linear program finds perfect matchings, let's suppose x_e can only take on integral values 0 or 1. Then, the equality constraint ensures that every vertex v has exactly one edge incident from it. This ensure that the edge set chosen is a matching.

Then, to motivate why it is a perfect matching, we look at a set S with an odd number of vertices, we know at least one of them is matched with some other node in $\bar{S}$, hence the second inequality constraint.

Edmonds showed that the above set of constraints define what is called the perfect matching polytope, where all extreme points of the polytope are integral and correspond to perfect matchings. This result is true for a general graph.

For a special type of graph that are cubic and 2-edge connected (i.e. every node has degree 3, and the graph is still connected after removing any edge), Naddef and Pulleyblank proved that $x_e = 1/3$ for all $e \in E$ is inside the feasible region.

Now if we consider the objective function $\Sigma_{e \in E} x_e$, by properties of extreme points, we know that at least one of the extreme points will have a better objective value than the $x_e = 1/3$ feasible point in the interior of the polytope. This implies that in any cubic 2-edge-connected graphs, there always exists a perfect matching of weight at least 1/3 of the total weight of edges.

Now we invoke Caratheodory's theorem which says that we can, in polynomial time, decompose a feasible solution in the perfect matching polytope into a linear combination of polynomially many perfect matchings. Suppose we regard the coefficients of this linear combination $\lambda_1, \lambda_2, \ldots$ as probabilities, and we choose a particular perfect matching M_i with probability λ_i, and combining with the above result by Naddef and Pulleyblank, we can assert that the probability of any edge being included in a perfect matching is exactly 1/3. We summarize this in the following lemma:

Lemma 2.2: Given any cubic, 2-edge-connected graph G, we can, in polynomial time, find a probability distribution over polynomially many perfect matchings such that the probability of any edge being chosen as part of a perfect matching is 1/3.

Notice that a 2-vertex-connected graph is automatically 2-edge-connected, unless it is the graph with 2 vertices and 2 edges connecting between them. Therefore, we can apply lemma 2.2 to cubic 2-vertex-connected graphs. Recall from lemma 2.1 that we can focus our attention to 2-vertex connected graphs, though in general they may not be cubic.

5 Main Idea of the Algorithm: Removable Pairs

With the preliminaries out of the way, we are now ready to describe the main features of this 1.461-approximate algorithm to graph-TSP.

Recall in Christofides algorithm, we first find the MST T of G, then "fix up" the odd degree nodes with a min weight matching M to get an Eulerian multigraph $H = T \cup M$.

Other algorithms for TSP employs a similar structure, except that instead of finding the minimum spanning tree, they use a minimum cost spanning 2-vertex-connected subgraph. But the use of matching is similar. Typically they find the minimum matching and add it to the 2-vertex-connected subgraph to arrive at an Eulerian multigraph. The main difference of this paper by Momke and Svensson is that instead of just adding edges from the matching M to the 2-vertex-connected subgraph, they showed that certain edges in M can instead be taken out of the 2-vertex-connected subgraph to yield a lower cost, while still maintaining the property that the resulting graph is Eulerian.

This prompts us to use the following definition, which may seem ad hoc now, but will become more clear later on.

Definition 3.1: (removable pairing of edges) Given a 2-vertex-connected graph G, we call a tuple (R,P) consisting of a subset R of removable edges and a subset $P \subseteq R \times R$ of pairs of edges a removable pairing if:

1. an edge is in at most one pair;

2. the edges in a pair are incident to a common vertex of degree at least 3;

3. any graph obtained by deleting removable edges so that at most one edge in each pair is deleted stays connected.

Now we are ready to prove the following lemma:

Lemma 3.3: Given a 2-vertex-connected graph $G = (V, E)$ with a removable pairing (R,P) defined above, we can, in polynomial time, find a probability distribution over polynomially many subsets of edges such that a random subset M form this distribution satisfies:

1. each edge is in M with probability $1/3$;

2. at most one edge in each pair is in M; and

3. each vertex has an even degree in the multigraph with edge set $E \cup M$.

Proof. We would like to transform G into G' which is a cubic 2-edge-connected graph, so that we can apply lemma 2.2. To do this, we transform every node that does not have a degree equal to 3. We will create a gadget for each type of node with degree not equal to 3.

Since G is 2-vertex-connected, clearly there can be no node of degree one, otherwise, we can just remove its only neighbor and it would be disconnected.

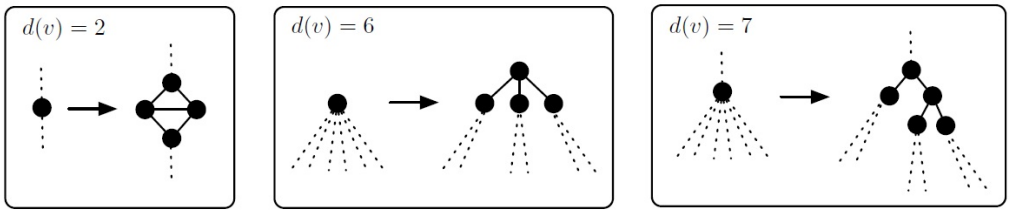

Figure 1: Examples of the used gadgets to obtain a cubic graph

1. For a vertex of degree 2, we replace that vertex with 4 new nodes as shown in the figure

2. for a vertex with degree $d(v) > 3$, we replace that vertex with a tree of $\lfloor \frac{d(v)}{2} \rfloor$ nodes. We ensure that every node inside the tree has degree 3. At each leaf node, we connect it to two edges originally in G such that if two edges $(e_1, e_2) \in P$, we make sure they get mapped to the same leaf node. When $d(v)$ is odd, there will be one extra edge remaining, we can fix this by connecting it to the root of the tree and making sure the root only has 2 children. Please see the graph for a few examples of this construction. Note that if we want to increase the degree by 2, we just simply split a leaf node and make it into two new leaf nodes while still satisfying the requirement that all nodes have degree 3.

Using these gadgets, we have turned G into G', which is 2-vertex-connected and cubic. This means we can apply lemma 2.2 to obtain a random perfect matching M'. We know each edge in G' will be included in M' with probability 1/3. Restricting the set M' to the graph G in the obvious way, we obtain M. Obviously, every edge in G will be in M with probability 1/3, satisfying the first requirement.

For the second requirement, suppose that both edges in a pair are in M. By the definition of a pair, they must be connected to a node of degree at least 3. That means, by the way we constructed the gadgets, the two edges in the pair will be connected to the same node in G', which means that they cannot both be in the matching M'. Hence, at mot one edge in each pair is in M.

For the third requirement, simply note that we are compressing the graph G' to get to G. $M' \cup E'$ is an Eulerian tour of G', which means that its nodes all have even degrees. The compression of a edge set with even degrees will

again be an edge set of even degrees. Therefore $M \cup E$ is a spanning Eulerian tour. Thus completing the proof.

We now use lemma 3.3 to prove a theorem which will be central to the rest of our analysis.

Theorem 3.2: Given a 2-vertex-connected graph $G = (V, E)$ with a removable pairing (R, P), there is a polynomial time algorithm that returns a spanning Eulerian multigraph in G with at most $\frac{4}{3}|E| - \frac{2}{3}|R|$ edges.

Proof: We begin by picking a random subset $M \subseteq E$ of edges that satisfies the properties of lemma 3.3. Let M_R be the set of those edges of M that are removable and let $\bar{M}_R$ be the set of the remaining edges of M.

Consider the multigraph $H = E \cup \bar{M}_R - M_R$, i.e. instead of adding all the edges in M into E, we remove the ones in the removable set. Since property 3 of the lemma ensures that $E \cup M$ has all even degrees, and add or removing an edge changes the parity of nodes in the same way, we must also have that H has all even degrees. Further, since (R, P) is a removable pairing, property 2 of lemma 3.3 ensures that H is connected. Therefore we can conclude that H is an Eulerian graph, i.e a solution to the graph-TSP problem.

Now we calculate the expected number of edges in H.

$$\mathcal{E}[|E| + |\bar{M}_R| - |M_R|] \tag{1}$$

Using Linearity of expectation, this can be evaluated to be:

$$|E| + \frac{1}{3}(|E| - |R|) - \frac{1}{3}|R| = \frac{4}{3}|E| - \frac{2}{3}|R| \tag{2}$$

Notice that in lemma 3.3, there are only a polynomial number of perfect matchings in the distribution, therefore we can simply check through all of them and take the one with the least number of edges, this gets rid of the randomization in the algorithm, and clearly, the choice with the least number of edges will have at most $\frac{4}{3}|E| - \frac{2}{3}|R|$ edges. This concludes the proof.

6 Motivation and Intuition

In Theorem 3.2, we showed that given a 2-vertex connected graph G with a removable pairing (R, P), we can find a Hamiltonian cycle with cost at most $\frac{4}{3} \cdot |E| - \frac{2}{3} \cdot |R|$. This bound is as yet unsatisfactory, because $|E|$ may be extremely large and $|R|$ may be small. What we'd like to do is find a *small*

spanning 2-vertex-connected subgraph G' of G with a large removable pairing (R, P) on G', such that Theorem 3.2's bound is tighter. We will show that we can find such a subgraph G' and a removable pairing (R, P) by solving a certain min-cost circulation problem.

7 Setup

We start with a spanning tree T of G which is obtained by a DFS, and use T to classify the edges of G as directed edges which are either **tree edges**, which are in T and directed towards T's leaves, and **back edges**, which are directed towards the root. Notably, there are no cross edges.

We extend G to obtain an integral min-cost circulation problem $C(G, T)$ with edge flow lower bounds. Every tree edge has a flow lower bound of 1, and G' is the set of all edges in G which correspond to positive-flow edges in $C(G, T)$, so G' will clearly span G. No other edge has a flow lower bound.

We would like to distinguish back-edges that come from different subtrees to make sure that the graph stays connected when edges are removed, so we introduce some new nodes. For every node v besides the root, we introduce new nodes $v_1, \ldots, v_l$ between v and each of its children. For such v , we redirect all of v's incoming back edges to the child v_i which lies between v and the other endpoint of the back edge. We will call these vertices *in-vertices*, and denote the set of in-vertices by $\mathcal{I}$.

All tree edges have cost 0, and all back edges have cost 1, with one caveat. Any given node can accept one unit of back-flow for 0 cost. This can be accomplished with standard min-cost circulation algorithms via an intermediate node with an edge of cost 0, capacity 1, and an edge of cost 1, infinite capacity.

Let $B(v)$ denote the set of back edges ending at v. Let $f(S)$ denote the amount of flow along the set of edges S.

The cost $c(C)$ is

$$\sum_{v \in \mathcal{I}} \max(f(B(v)) - 1, 0)$$

Now we give what the removable pairing (R, P) is, but we will postpone actually showing that it is a removable pairing.

We will define R and P on $C(G, T)$, but they reduce to a removable pairing on G' quite easily. P consists of all pairs of zero-cost back edges and tree edges which meet at an in-vertex with degree at least three. Note

that the restriction of having degree at least three will only ever exclude the root node, because every other in-vertex has an in-bound tree edge, and out-bound tree edge, and we have already hypothesized that it has an in-bound back edge. Since each in-vertex only has one inbound zero-cost back edge, all zero-cost back edges except possibly one to the root are included in P.

R is P union all other back edges. So $R - P$ is the set of all positive-cost back edges, and possibly one zero-cost back edge to the root. This implies that $|R| - 2|P| \leq c(C) + 1$. Furthermore, the number of edges in G' is the tree edges + the positive-flow back edges, which is at most $(n-1)+|R|-|P|$ (we need to count the zero-cost edges).

This means, once we show that (R, P) meets the criteria for being a removable pairing, Theorem 3.2 will imply that G' has a spanning Eulerian multigraph with at most $\frac{4}{3}((n-1)+|R|-|P|)-\frac{2}{3}|R| = \frac{4}{3}n+\frac{2}{3}(|R|-2|P|) \leq \frac{4}{3}n + \frac{2}{3}c(C^*) - \frac{2}{3}$ edges.

We now need to show that (R, P) is a removable pairing.

First, an edge is in at most one pair of P. P was defined as all pairs of zero-cost back edges and tree edges which meet at an in-vertex with degree at leaset three. Zero-cost back edges and tree edges are uniquely defined by the in-vertex that they touch, so it is clear that all these pairs will be disjoint from each other. When we reduce to G' from $C(G, T)$, neither of these two edges are part of the

Second, the edges in a pair are incident to a common vertex of degree at least 3. This is by construction of P. When we reduce to G' from $C(G, T)$, the degree of the edges' common node will still be at least 3, because they will still have an in-bound tree edge.

Finally, we need to show that if we remove edges in R such that we remove no two edges from the same pair in P, the graph stays connected. To prove this, we use induction on the height of a subtree. For leaves, of course removing edges will not disconnect a leaf vertex from itself. For a subtree of height h, by induction each of its sub-trees remains connected, and we just need to show that the root v of this subtree remains connected to each subtree. Suppose that the edge between v and one of its children has been deleted. Since this is a tree edge, the other element in its pair must have been a back-edge from this child's subtree to v, which cannot have been deleted, so v remains connected.

Examples of transformations go here

8 Bounding the Circulation

Recall that for a graph G, $LP(G)$ refers to the Held-Karp linear program for G, in which variables correspond to fractional edge inclusion, and the fractional magnitude of each non-trivial cut is at least 2. It is a known fact (we will not prove it) that an extreme point of $LP(G)$ has at most $2n-1$ non-zero variables. We call this set of non-zero variables the **support** of the extreme point, and we call the corresponding set of edges E'. We use $OPT(LP(G))$ primarily as a lower bound on the optimum path in order to prove approximation ratios.

That is, we say we have an r-approximation relative to $LP(G)$ if our solution's magnitude is within a factor of r of $OPT(LP(G))$. An r-approximation relative to $LP(G)$ is also just an r-approximation, because $OPT(LP(G))$ is less than or equal to the optimal integral solution.

Now, consider the graph $G = (V, E')$. $OPT(LP(G')) = OPT(LP(G))$ because by construction, E' corresponds to a small set of edge variables which are sufficient to achieve $OPT(LP(G))$. So we just need to be able to find an r-approximation for graphs with n vertices and at most $2n-1$ edges.

Let x_a denote the value corresponding to edge a in $LP(G')$. For reasons that will become apparent in our analysis, we choose a tree T for G' with a DFS from an arbitrary node, but when we have a choice of which edge to traverse first, we pick the edge with the highest corresponding value in $LP(G')$).

Now to bound the minimum circulation in $C(G', T)$, we give an example circulation with a boundable value. Our circulation f consists of two parts, which we will call f' and f''. f' consists of, for each back edge e, sending $\min(x_e, 1)$ unit of flow along the (unique) cycle containing only e and tree edges.

For this circulation to be feasible, we also need to ensure that each tree edge has flow at least 1. This is what f'' does. Because from an in-vertex, there is only one place for flow to go (along the outbound tree edge), it is sufficient to consider to require only that the flow going into each in-vertex along a tree edge is at least 1. For each tree edge (v, w), where v is an out-vertex, and w is a child of v, $f'(v, w)$ is less than 1, f'' sends $1 - f'(v, w)$ flow along some cycle containing (v, w) and a back edge. There must be such a cycle, because otherwise we could show that 2-vertex-connectedness is violated.

Now we analyze the cost of this circulation.

It is equal to

$$\sum_{v \in \mathcal{I}} \max(f(B(v)) - 1, 0)$$

, which is upper-bounded by

$$\sum_{v \in \mathcal{I}} \max(f'(B(v)) - 1, 0) + \sum_{vin\mathcal{I}} f''(B(v))$$

I claim that $\sum_{vin\mathcal{I}} f''(B(v)) \leq OPT(LP(G)) - n$. Also, $\sum_{vin\mathcal{I}} \max(f'(B(v)) - 1, 0) \leq (7 - 6\sqrt{2})n + 4(\sqrt{2} - 1)OPT(LP(G))$. However, I will not prove either of these.

Combining the two gives us that the cost of the circulation is upper-bounded by $6(1 - \sqrt{2})n + (4\sqrt{2} - 3)OPT(LP(G))$.

On the other hand, Christofides' algorithm can also be analyzed with an approximation ratio in terms of $OPT(LP(G))$, and its approximation ratio is $\dfrac{n + OPT(LP(G))/2}{OPT(LP(G))}$.

By solving $LP(G)$, we can pick whichever would perform better, and achieve an approximation ratio of $\dfrac{14(\sqrt{2} - 11)}{12\sqrt{2} - 13}$, which is about 1.461.