Available online at www.sciencedirect.com

SCIENCE DIRECT®

Operations Research Letters 33 (2005) 467–474

Operations Research Letters

www.elsevier.com/locate/orl

An improved upper bound for the TSP in cubic 3-edge-connected graphs

David Gamarnik[b], Moshe Lewenstein[a,*,1], Maxim Sviridenko[b]

[a]*Department of Computer Science, Bar Ilan University, Room 323, Mailbox 50, Ramat Gan 52900, Israel*
[b]*IBM T. J. Watson Research Center, Yorktown Heights, P.O. Box 218, NY 10598, USA*

Received 20 January 2004; accepted 3 September 2004

Abstract

We consider the travelling salesman problem (TSP) problem on (the metric completion of) 3-edge-connected cubic graphs. These graphs are interesting because of the connection between their optimal solutions and the subtour elimination LP relaxation. Our main result is an approximation algorithm better than the 3/2-approximation algorithm for TSP in general.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Approximation algorithms; Travelling salesman problem; Graphs; Regular graphs

1. Introduction

In this paper we consider the well-known *travelling salesman problem* (*TSP*). Given a complete undirected graph $G_c = (V, E_c)$ with nonnegative edge weights, the goal is to find a shortest (according to the weights) closed tour visiting each vertex exactly once. A closed tour visiting each vertex exactly once is also known as a *Hamiltonian cycle*.

In general the TSP problem is NP-complete and not approximable to any constant. Even the metric variant of TSP, where the edge weights of G_c form a metric over the vertex set, is known to be MAXSNP-hard [8]. However, Christofides [5] introduced an elegant approximation algorithm for metric TSP that produces a solution with an approximation factor 3/2. Almost three decades later, there is still no approximation algorithm with a better factor than Christofides' 3/2 factor. A detailed survey of the results and the history of TSP appears in [7,6].

Let G be an arbitrary graph with edge weights that do not violate the triangle inequality. A *metric completion* on G is the completion of G into a complete graph G_c such that each added edge (u, v) is weighted in a manner so that G_c forms a metric. A *shortest path metric completion* is a metric completion for which

* Corresponding author. Tel.: +972 3 531 7668; fax: +972 3 736 0498.

E-mail addresses: gamarnik@watson.ibm.com (D. Gamarnik), moshe@cs.biu.ac.il (M. Lewenstein), sviri@us.ibm.com (M. Sviridenko).

[1] The research in this paper was done when the author was a postdoc at IBM T.J. Watson Research Center.

each edge added is assigned the weight of the shortest path between u and v. Since we only consider shortest path metric completions in this paper, we will refer to them as metric completions. In general, in this paper we study the case when G is a cubic 3-edge connected graph and G_c is its metric completion. All edge weights of the original graph are 1.

One of the classical integer programming formulations of the TSP problem is as follows:

$$\min \sum_{(u,v)\in E_c} w_{(u,v)} x_{(u,v)} \tag{1}$$

$$\sum_{(u,v)\in E_c} x_{(u,v)} = 2 \quad \text{for all } v \in V, \tag{2}$$

$$\sum_{u\in S, v\in V\setminus S} x_{(u,v)} \geqslant 2 \quad \text{for all } \emptyset \neq S \subset V, \tag{3}$$

$$x_{(u,v)} \in \{0,1\} \quad \text{for all } u, v \in V. \tag{4}$$

We denote optimal solutions of the above integer program with IP^* and its value with ρ_{IP^*}. As mentioned above, finding IP^* is NP-complete. Changing requirement (4) to require that $x_{(u,v)} \geqslant 0$ yields a linear programming relaxation of the integer program also known as the *subtour elimination relaxation* (SER for short).

Despite the drawback that there are an exponential number of constraints (of type (3)), SER can nevertheless be solved in polynomial time since each iteration of the separation oracle is one mincut computation. We denote with LP^* the solution to SER and with ρ_{LP^*} its value. A measure for the quality of approximation of an SER solution is the worst-case ratio (integrality gap)

$$\rho = \max \frac{\rho_{APX}}{\rho_{LP^*}},$$

where ρ_{APX} is a value of an SER solution and the maximum is over all instances of metric TSP.

The best-known upper bound to date [10,11] shows that $\rho \leqslant 3/2$. However, no example showing ρ to be 3/2 is known. The well-known, and currently best, lower bound on ρ appears in Fig. 1. The full example is a metric completion on this graph. To see the 4/3 gap, set $x_{(u,v)}$ to be 1/2 for the weight 2 edges in the graph and $x_{(u,v)}$ to be 1 for the weight 1 edges. For all other edges, set $x_{(u,v)} = 0$. In Fig. 1 the optimal tour is shown in the right-hand figure.

This leads to the following conjecture with regard to ρ.

Conjecture. *For metric TSP, the integrality gap ρ for SER is* 4/3.

There have been numerous attempts to prove this conjecture true; however, so far these attempts have been unsuccessful. In fact, not only has the conjecture not been proven so far, there has not even been an improvement over the 3/2 factor. This has raised the counterquestion: perhaps 3/2 is the best factor achievable for the metric TSP problem. Or a somewhat milder variant of the question is, perhaps for $\rho = \rho_{IP^*}/\rho_{LP^*}$, $\rho = 3/2$ is the best achievable.

An optimal solution to the subtour elimination LP is closely related to k-regular k-edge-connected multigraphs in the following sense. Given an optimal solution to the LP, we define the number D to be the smallest common multiplier of the variables x_e^* for all edges e, i.e. D satisfies that for each edge e, Dx_e^* is integer. On the vertex set V we define a multigraph with Dx_e^* edges between vertices u and v where $e = (u, v)$. This multigraph is $2D$-regular and $2D$-edge-connected by the constraints of the LP. Consider the original graph G, which was later completed to a metric as a TSP instance. Say the weights of all edges were 1, i.e. it was unweighted. If the optimal value of the LP was n, i.e. G was "fractionally" Hamiltonian, then it would be enough to find a tour of value αn with $\alpha < 3/2$ to improve on Cristofides algorithm for that special (yet very general) case.

We consider probably the simplest class of graphs satisfying the above properties, the cubic (that is, 3-regular) 3-edge-connected simple graphs with weights 1. The metric completions of these graphs have weights that can be substantially large. It is easily seen that $\rho_{LP^*} = n$ on this class of graphs. Indeed, consider the following solution: $x_{(u,v)} = 2/3$ if $(u, v) \in E$ and $x_{(u,v)} = 0$ otherwise. The feasibility of this solution follows from the fact that graph G is cubic and 3-edge-connected. Since the value of this solution is n and n is a trivial lower bound on ρ_{LP^*}, this solution is optimal.

Thus, to improve the bound on the value of the optimal Hamiltonian cycle, it is enough to prove that there is a Hamiltonian cycle of weight at most αn in G_c for $\alpha < 3/2$. In this paper, we design a polynomial

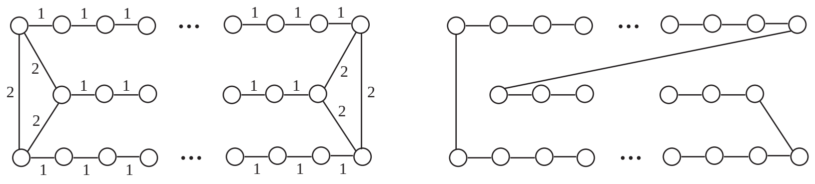

Fig. 1. An example for which ρ is 4/3.

time algorithm which finds a Hamiltonian cycle in G_c of weight at most $(3/2 - 5/389)n$. Approximation algorithms with performance guarantees better than $3/2$ (or even PTASes) for other special classes of metric spaces can be found in [1–3,8].

There are several directions for possible improvements of our results. It is conceivable that the improvements over $3/2$ approximations can be obtained for general r-regular graphs. Another possibility is to obtain improved results for sparse graphs, say graphs with maximum degree 3. At this point, we cannot even lift the 3-connectivity restriction.

2. Preliminaries

Consider a cubic 3-edge-connected graph G and the corresponding complete weighted graph G_c. Any Hamiltonian cycle in G_c corresponds to some closed path in G of the same weight that visits all vertices at least once and, conversely, any such path corresponds to a Hamiltonian cycle in G_c of the same weight. Therefore, our original problem of finding a shortest Hamiltonian cycle in G_c can be reformulated as the problem of finding an Eulerian multigraph with a minimum number of edges on the vertex set V using edges from G only (possibly more than once).

Let $G = (V, E)$ be a graph and $E' \subset E$ be a collection of edges. Let $V_1, \ldots, V_k$ be a partition of the vertices according to the connected components of (V, E'). The *contraction* of G according to E' is a multigraph $G^* = (V^*, E^*)$, where $V^* = \{V_1, \ldots, V_k\}$ and E^* contains an edge (V_i, V_j) for each original edge $(u, v) \notin E'$ where $u \in V_i$ and $v \in V_j$. Note that there is a one–one correspondence between the edges of E' from G and the edges E^* from G^*. Also note that there may be parallel edges in G^*.

Also say we have a spanning tree $T = (V, E)$ on a connected graph G. We call the multigraph (V, E'), where E' contains two copies of each edge of E, a *double spanning tree*. We say that we *double T* when we take a double spanning tree according to T.

A *cycle cover* is a collection of node disjoint cycles covering all nodes in a graph. The following result from 1891 is due to Petersen [9]. Its proof can be found in Behzad, Chartrand and Lesniak-Foster [4].

Lemma 1 (*Petersen [9]*). *Let $G=(V, E)$ be a 2-edge-connected cubic graph. The edge set E can be partitioned into a perfect matching M and a cycle cover C.*

Note that both problems of finding a perfect matching and finding a cycle cover are polynomially solvable. Using this result we show the following lemma.

Lemma 2. *Let $G=(V, E)$ be a 3-edge-connected cubic graph. The edge set E can be partitioned into a perfect matching M and a triangle-free cycle cover C (that is a cycle cover which does not contain cycles of length 3).*

Proof. We prove the lemma by induction on the size of the graph G. If $|V|=4$ the lemma is trivial. Assume we proved it for all graphs G with $|V| \leqslant n-2$ (note that $|V|$ must be even). Consider a graph G with $|V|=n$. If G does not contain triangles, then by Lemma 1 graph G can be split into a cycle cover C (which must be triangle-free) and a matching M. Otherwise, let E' be an arbitrary triangle and let G^* be the contraction of G according to E'. G^* is still cubic and 3-edge-connected and has $n-2$ vertices, and therefore can be split into a matching and a triangle-free cycle cover C. Note that in general when contracting to G^*, parallel edges, and hence loops, may be created. However, since G is 3-edge-connected this does not happen. The contracted triangle E' belongs to some cycle of length at least $l \geqslant 4$ in C. If we uncontract E', we can add two edges from E' to the cycle and get a cycle of length $l+2$ in the original graph G. The remaining edge of the triangle E' is added to the matching. □

3. The TSP approximation algorithm

3.1. Algorithm outline

The algorithm we present utilizes the partition of the edge set E into $M \cup C$ implied by Lemma 2. (We will be interested in C only and not in M.) C, the cycle cover, is triangle-free. Hence, each cycle in C is of length at least four. Obviously, if there is exactly one cycle in the cycle cover, this is a Hamiltonian cycle and we are done. Likewise, if all the cycles are relatively large then transforming it into a Hamiltonian cycle can be done at a small “cost”. The algorithm considers two cases. In the first a large fraction of the cycles are of size 5 or greater (we call this the high-five property—to be defined later). If the cycle cover C is high-five then we can benefit from the fact that there are many large (greater than length four) cycles and utilize this to find a Hamiltonian cycle at a “reduced cost”.

The more difficult situation to handle is when the cycle cover C is not high-five. Here C contains a lot of cycles of length four. In this case we manipulate C in a collection of rounds in order to transform it into a Eulerian graph from which a Eulerian tour is taken. This is the heart of the algorithm and can be done in a manner that improves over the approximation factor over $3/2$.

3.2. The high-five property

Throughout the algorithm presentation we will work with a constant ε, to be set at the end of the analysis. Note that, as mentioned above, the larger the cycles in C the closer they are to a “real” TSP. The following definition captures the notion of having many larger cycles.

The high-five property: We say that a cycle cover has the *high-five* property if the cycles of length five and more in C contain at least εn vertices, where n is the number of vertices in V.

Let us choose a partition of the edge set $E = M \cup C$ according to Lemma 2. We will be interested in the cycle cover C. We first consider the situation when C is high-five and then the, more difficult case, when C is not high-five. Finally we choose the ε as a suitable tradeoff between the high-five and non high-five cycle covers as a result we will prove the following theorem.

Theorem 3. *Given an arbitrary cubic 3-edge-connected graph, the metric completion of this graph contains a Hamiltonian cycle of weight at most* $(3/2 - 5/389)n$.

3.3. High-five cycle covers

When the cycle cover C is high-five, it is straightforward to show that an improvement over the $3/2$ Christofides’ approximation can be obtained. This improvement is dependent on ε and is as follows.

Lemma 4. *If the cycle cover C is high-five, then G_c has a Hamiltonian cycle of weight at most* $(3/2 - \varepsilon/10)n$.

Proof. Let G' be the contraction of G according to C. Let T be a spanning tree on G'. Double T to get T'. We output $G'' = (V, C \cup T')$. Clearly G'' is Eulerian since it is connected and all vertices have even degree and therefore it corresponds to a Hamiltonian cycle in G_c.

We now estimate the number of edges in G''. Let $V_4 \subseteq V$ be the set of vertices covered by the cycles of length 4 in C. Since C satisfies the high-five property, $|V \setminus V_4| \geqslant \varepsilon n$ and therefore the total weight of the tour is estimated above by

$$n + 2\left(\frac{|V_4|}{4} + \frac{|V \setminus V_4|}{5}\right) \leqslant \left(\frac{3}{2} - \frac{\varepsilon}{10}\right)n. \quad \square$$

3.4. Cycle covers that are not high-five

We now consider the case where the cycle cover C is not high-five, i.e. when $|V \setminus V_4| < \varepsilon n$. Let C' be the edges of C from cycles of length exactly 4. Let G' be the contraction of G according to C'. First of all we claim that G' does not contain self-loops since otherwise it would contain the subgraph from Fig. 2 and therefore would not be 3-edge-connected. So G' contains $|V_4|/4$ vertices of degree four and $|V \setminus V_4| < \varepsilon n$

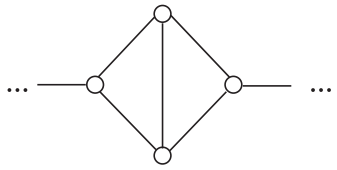
Fig. 2. Impossible 4-cycle in a 3-edge-connected graph.

vertices of degree three. The total number of edges in G' is $(|V_4| + 3|V \setminus V_4|)/2$.

3.4.1. Eliminating isolated 2-cycles

In the contracted graph G' there are nodes that correspond to a single node in G and hence still have degree 3, and nodes that correspond to cycles of length 4 in C the cycle cover on G. It is easy to verify that these nodes have degree 4. Moreover we may have parallel edges in G'.

A 2-cycle is a graph consisting of two parallel edges. We call a 2-cycle isolated if it does not have a common vertex with another 2-cycle. For each isolated 2-cycle in G', delete one of the two parallel edges in the 2-cycle (notice that we cannot have three parallel edges in G' since otherwise G would not be 3-edge-connected). Let G'' be the new graph. Since isolated 2-cycles do not touch each other the deleted edges form a matching in G' and therefore the total number of edges in G'' is at least

$$\frac{|V_4| + 3|V \setminus V_4|}{2} - \frac{|V_4|/4 + |V \setminus V_4|}{2} = \frac{3|V_4|}{8} + |V \setminus V_4|.$$

3.4.2. Removing 2-cycle chains and cycles

It is possible that in G'' some 2-cycles share a node. However, no more than two 2-cycles may share a node, since the degree of vertices in G'' is at most 4. In fact, a node incident on two 2-cycles has no other edges incident upon it. Such a node is said to be *2-cycle-saturated*. Since each 2-cycle saturated node is incident to exactly two 2-cycles, the 2-cycle-saturated nodes form a collection of (disjoint) chains (of 2-cycles) and cycles (of 2-cycles). We also call the 2-cycle-saturated nodes, *internal nodes* as they must be internal to the chain or cycle. However, note that the only cycle of 2-cycles possible is one that contains all vertices. This is true since all nodes on this cycle

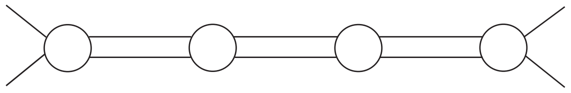
Fig. 3. A chain of 2-cycles.

are 2-cycle-saturated and hence not incident to any other edges, implying that the cycle of 2-cycles is a component of G''. However, G'' is a connected graph (Fig. 3).

Denote with C'' all 2-cycles on chains. For each chain the 2 vertices, at either end, are not 2-cycle saturated and hence are incident to one or two other edges, the *tail edges*. We remove the C'' and tail edges from G''. In the case where there is exactly one cycle of 2-cycles, we let C'' be a Hamiltonian cycle containing one edge from each 2-cycle. In this case we remove the Hamiltonian path from G''. After removing these edges, G'' does not contain 2-cycles anymore.

We now (greedily) find any cycle in G'', delete its edges and all edges incident to this cycle from the graph and add the cycle to C''. We repeat this process until the graph G'' becomes acyclic.

Let p be the number of steps (cycles and chains) in the above process and L_i, $i = 1, \ldots, p$, be the number of vertices in the chain or cycle defined in step i. At the end of the above algorithm, G'' contains at most $|V_4|/4 + |V \setminus V_4| - \sum_{k=1}^{p} L_k$ vertices that are not isolated. Since G'' is acyclic without parallel edges, i.e. a forest, there are at most $|V_4|/4 + |V \setminus V_4| - \sum_{k=1}^{p} L_k - 1$ edges. For each cycle or chain of length L_k deleted from G'', there are at most $3L_k$ edges deleted from G'' and therefore

$$\begin{aligned} 3\sum_{k=1}^{p} L_k &\geqslant \frac{3|V_4|}{8} + |V \setminus V_4| \\ &\quad - \left(|V_4|/4 + |V \setminus V_4| - \sum_{k=1}^{p} L_k - 1\right) \\ &\geqslant \frac{|V_4|}{8} + \sum_{k=1}^{p} L_k. \end{aligned}$$

Hence,

$$\sum_{k=1}^{p} L_k \geqslant \frac{|V_4|}{16}. \tag{5}$$

Now, consider the original G'', i.e. the graph we had before removing the edges according to C''. Let G''' be the contraction of G'' according to C''. Remove the self-loops from G''' and let T be a spanning tree in G'''.

Let us estimate the total number of vertices in G''', which is one more than the size of the spanning tree T. This number is also equal to the number of vertices in G'' minus the total number of vertices in the cycles and chains from C'' and plus the number of cycles and chains in C'', i.e.

$$\begin{aligned}|T|+1 &\leqslant |V_4|/4+|V\setminus V_4|-\sum_{k=1}^{p} L_k+p\\ &\leqslant |V_4|/4+|V\setminus V_4|-\frac{2\sum_{k=1}^{p} L_k}{3},\end{aligned} \tag{6}$$

where $|T|$ denotes the number of edges in tree T and the second inequality follows from the fact that $L_k \geqslant 3$ for all $k=1,\ldots,p$.

3.4.3. *Final Eulerian graph*

We now build the final Eulerian graph in G. Let C''_G be the edges in the graph G corresponding to the edges from C'' (note that even though we have been transforming G, the edges in every stage are from the original graph G, so although C'' was defined on a transformed graph the edges are still originally from G). We will transform the graph $C \cup C''_G$ so that the degrees of all vertices will become even to satisfy the Eulerian requirements. We also guarantee that the new graph contains at most $n+\sum_{k=1}^{p} L_k$ edges and that the connectivity properties are not destroyed, i.e. every connected component of the transformed graph is a connected component of the original graph and vice versa. After we transform $C\cup C''_G$ to maintain the "even degree" requirement we add a doubled spanning tree T to connect all connected components of $C\cup C''_G$ while maintaining the "even degree" requirement. Hence, the graph will be Eulerian.

3.4.4. *Transforming $C \cup C''_G$ into a graph with even degree vertices*

We begin by analyzing the reasons for an odd degree vertex in $C \cup C''_G$. Since C is a cycle cover on the original G, each vertex has exactly two edges in $C\cup C''_G$ from the C cycles. G'' is obtained by contracting cycles of length 4 in C into (super)vertices in G''

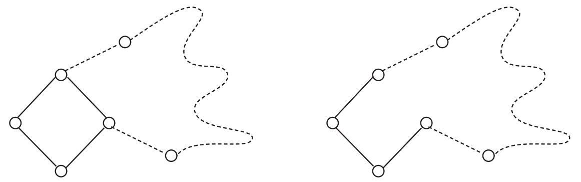
Fig. 4. Solid edges are the 4-cycle of C and the dashed edges are from the cycle in G''. Note that the 4-cycle is a supervertex in G''.

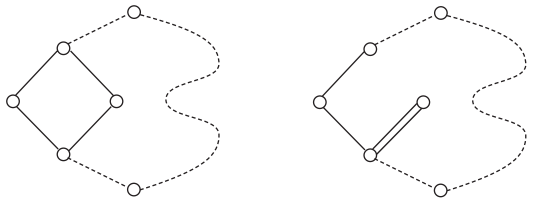
Fig. 5. Solid edges are the 4-cycle of C and the dashed edges are from the cycle in G''.

(and possibly removing edges from isolated 2-cycles). Since C'' contains cycles and chains of 2-cycles in G'', a vertex v in G can be adjacent to either 0, 1 or 2 edges from C''_G. The parity of the degree of v is odd if it is adjacent to exactly one edge from C''_G. This can happen only if v belongs to a 4-cycle in C. Moreover by construction of C'' the cycles and chains of 2-cycles in C'' are vertex disjoint (relative even to G''), so for each 4-cycle in C, which corresponds to a vertex in G'', it can be either on a cycle of C'', on a 2-cycle chain or on neither. But never on both. Hence we have two cases.

Case 1: This vertex belongs to some cycle of length four and it is incident to exactly one edge from some cycle in C''_G (see left figures in Figs. 4 and 5).

For any cycle of length four in C there are either none or two such vertices. In the latter case, if these vertices are adjacent in the cycle of length four we just delete the edge connecting them which decreases their degrees by one, see Fig. 4. This procedure does not change connectivity since we throw away just one edge per cycle. If a cycle v_1, v_2, v_3, v_4 of length four has two vertices v_1, v_3 of odd degrees which are not adjacent we throw away edge (v_1, v_2) and add edge (v_2, v_3), see Fig. 5. Again after this transformation the degrees of all vertices in the cycle are even and the connectivity of the component remains intact. Moreover, the transformation does not increase the total number of edges in a component.

Case 2: The vertex of odd degree belongs to a cycle of length four and the vertex corresponding to this cycle in G'' belongs to a chain of 2-cycles in C''.

Lemma 5. *Every node in this chain corresponds to a cycle of length four in C.*

Proof. Indeed, every internal (2-cycle-saturated) node of this chain is a cycle of length four in C since it has degree four in G''. Every end node of a chain also corresponds to a 4-cycle. In fact, otherwise it would have degree three in G'' and two edges of a chain connecting this vertex to an internal 4-cycle (which belongs to C) cannot belong to any cycle in C. Since the end node itself does belong to some cycle in C, it must be incident to two additional edges, but then its degree is 4, a contradiction. □

We distinguish between internal (2-cycle-saturated) nodes on these chains and between the end nodes of the chain. Consider a 4-cycle corresponding to an internal (2-cycle saturated) node of a chain. Let us denote it v_1, v_2, v_3, v_4. All these vertices have degree three in $C \cup C''_G$, 2 edges from the cycle C that they participate upon and the only other edge v_i is connected to corresponds to a 2-cycle edge in G'' which is chosen in C'' as a chain of 2-cycles (since this 4-cycle is an internal node in G'').

Therefore, by deleting two edges $(v_1, v_2), (v_3, v_4)$ or $(v_1, v_4), (v_2, v_3)$ we make all degrees of vertices v_1, v_2, v_3, v_4 even. Moreover, one of these two variants does not destroy connectivity, since the initial chain consisting of the internal 4-cycles, end 4-cycles and pairs of edges connecting the 4-cycles, contains two vertex disjoint paths, in G, connecting end cycles. Therefore, if we throw away two opposite edges in each internal 4-cycle not belonging to these paths we do not disconnect vertices of the 4-cycle from each other. For two external nodes of a chain we apply the argument identical to the one from case 1. There are exactly two nodes of odd degree on a 4-cycle corresponding to an end of a chain; if these two nodes are connected by an edge in a 4-cycle, we delete it. Otherwise, if v_1 and v_3 are vertices with odd degree in a 4-cycle v_1, v_2, v_3, v_4 we delete the edge (v_1, v_2) and add the edge (v_2, v_3).

3.4.5. Upper bounding the size of the Eulerian graph

In the previous section we explained how to transform graph $C \cup C''_G$ into an Eulerian graph. Let us call this graph G_t. We now upper bound the total number of edges in G_t. If p_1 is the number of chains in C'' and p_2 is the number of cycles in C'', then $C \cup C''_G$ contain at most $n + 2\sum_{k=1}^{p_1}(L_k - 1) + \sum_{k=p_1+1}^{p} L_k$ edges where $p = p_1 + p_2$. During the transformation we never increase the current number of edges in our graph, but we need a stronger statement since we would like to show that the number of edges in G_t is at most $n + \sum_{k=1}^{p} L_k$; we do it by showing that we delete enough edges when we transform chains of supervertices corresponding to cycles of length four. Indeed, for each chain of length L_k we delete at least $2(L_k - 2)$ edges when we transform internal cycles of a chain (2 per each internal node of a chain). Therefore, the total number of edges in G_t can be upper bounded by

$$
\begin{aligned}
& n + 2\sum_{k=1}^{p_1}(L_k - 1) + \sum_{k=p_1+1}^{p} L_k - 2\sum_{k=1}^{p_1}(L_k - 2) \\
& \quad \leqslant n + 2p_1 + \sum_{k=p_1+1}^{p} L_k \leqslant n + \sum_{k=1}^{p} L_k
\end{aligned}
$$

since $L_k \geqslant 3$ for all k.

Let us estimate the total number of edges in the final Eulerian graph, which contains the G_t and a doubled spanning tree T over its connected components. This number can be estimated above by

$$
\begin{aligned}
& n + \sum_{k=1}^{p} L_k + 2|T| \\
& \quad \leqslant n + |V_4|/2 + 2|V \setminus V_4| - \frac{\sum_{k=1}^{p} L_k}{3} \\
& \quad \leqslant n + \left(\frac{1}{2} - \frac{1}{48}\right)|V_4| + 2|V \setminus V_4| \\
& \quad \leqslant \left(\frac{3}{2} - \frac{1}{48}\right) n + \left(\frac{3}{2} + \frac{1}{48}\right) \varepsilon n, \qquad (7)
\end{aligned}
$$

where the first inequality follows from (6), the second one follows from (5) and the last one follows from the fact that $|V \setminus V_4| < \varepsilon n$. The inequality (7) implies the following lemma.

Lemma 6. *If the cycle cover C is not high-five, then G_c has a Hamiltonian cycle of weight at most* $(3/2 - 1/48)n + (3/2 + 1/48)\varepsilon n$.

3.5. Trading off high-five and non high-five cycle covers

To achieve our final result, we need to tradeoff high-five cycle covers and cycle covers which are not high-five. To achieve this we optimize ε over inequalities in Lemmas 4 and 6 and obtain Theorem 3.

The construction is obviously achievable in polynomial time, since all it involves is finding a perfect matching (the cycle cover is then immediately obtainable), detecting cycles of length 4, chains of 2-cycles over them and then greedily finding cycles. Also the spanning tree can be found in polynomial time.

Acknowledgements

We thank the anonymous referees of this paper for providing comments that improved the exposition of this paper.

References

[1] S. Arora, Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems, J. ACM 45 (5) (1998) 753–782.

[2] S. Arora, M. Grigni, D. Karger, P. Klein, A. Woloszyn, A polynomial-time approximation scheme for weighted planar graph TSP, Proceedings of the Symposium on Discrete Algorithms, 1998, pp. 33–41.

[3] D. Arun Kumar, C. Pandu Rangan, Approximation algorithms for the traveling salesman problem with range condition, Theoret. Inform. Appl. 34 (3) (2000) 173–181.

[4] M. Behzad, G. Chartrand, L. Lesniak-Foster, Graphs and Digraphs, PWS Publishers, Massachusetts, 1979.

[5] N. Christofides, Worst-case analysis of a new heuristic for the travelling salesman problem, Technical Report, Carnegie Mellon University, 1976.

[6] G. Gutin, A. Punnen (Eds.), The Traveling Salesman Problem and its Variations, Kluwer Academic Publishers, Dordrecht, 2002.

[7] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (Eds.), The Traveling Salesman Problem, John Wiley, New York, 1985.

[8] C.H. Papadimitriou, M. Yannakakis, The traveling salesman problem with distances one and two, Math. Oper. Res. 18 (1993) 1–11.

[9] J. Petersen, Die Theorie der Regulären Graphen, Acta Math. 15 (1891) 193–220.

[10] D.B. Shmoys, D.P. Williamson, Analyzing the Held-Karp TSP bound: a monotonicity property with application, Inform. Process. Lett. 35 (6) (1990) 281–285.

[11] L. Wolsey, Heuristic analysis, linear programming and branch and bound, Math. Programming Stud. 13 (1980) 121–134.