Towards a 4/3 Approximation for the Metric Traveling Salesman Problem

SWATI GUPTA

Department of Computer Science And Engineering

Indian Institute of Technology, Delhi

12th May 2011

Towards a 4/3 Approximation for the Metric Traveling Salesman Problem

A thesis submitted in partial fulfilment of

the requirements for dual degree

(Bachelor and Master of Technology)

in

Computer Science and Engineering

by

SWATI GUPTA 2006CS50451

under the guidance of

Prof. NAVEEN GARG

Department of Computer Science And Engineering

Indian Institute of Technology, Delhi

12th May 2011

Certificate

This is to certify that **Swati Gupta** has worked under my supervision for her Master of Technology thesis entitled *Towards a 4/3 Approximation for the Metric Traveling Salesman Problem* during the course of her dual degree (Bachelor and Master of Technology) at Indian Institute of Technology, Delhi. This work is a bonafide record of research work carried out by her under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma. All sources and references have been acknowledged.
I hereby recommend this submission to the institute in partial fulfilment of the requirements for dual degree (Bachelor and Master of Technology) in Computer Science and Engineering.

Professor Naveen Garg
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

Acknowledgments

"'Unsolved problems give us an occasion to fight the ignorances of human mind"' - Prof. Ram Murty of Queen's University.

Introducing me to the wonderful Traveling Salesman Problem has been one of the best things Prof. Naveen Garg, my advisor, has done for me and now it has become nothing short of a passion for me. I remember telling Prof. Naveen Garg that this problem was very hard and that I was scared of pursuing it as my thesis project. But it is only with his constant encouragement, belief and support that I have been able to write this thesis today. I will always value our discussions on this project, for some of his statements have been defining ones for me. His clarity in thought and ideas has helped me not only progress through the project but also be a less confused person.

Secondly, I would like to thank Prof. Amitabha Tripathi for instilling in me a fondness for Graph Theory. I took this class in my seventh semester - scared almost by his reputation of failing half the students. But after taking it for a semester and rigorous problem solving I was sure that I wanted to work on a combination of Graph Theory and Algorithms in the coming years. He guided

my proof writing and taught me how to write beautiful documents using latex. He also showed us a documentary of Fermat's Last Theorem - that made me realise that it might just be awesome to work on a problem for seven years.

I would also like to thank Nishita Aggarwal, who enthusiastically worked with me during the later part of the thesis. She listened to me, asked innocent doubts that were sometimes loopholes in my proofs and helped in removing them. Discussions with her helped me clear my head, structure my thinking, and focus on one idea at a time. She also helped me in drawing some pictures used in this thesis.

It was not only working on this problem that kept me busy but the interesting course work in Approximation Algorithms for which I'm grateful to Prof. Naveen Garg and Prof. Amit Kumar and Advanced Algorithms - for I learnt the love of teaching from Prof. S.N.Maheshwari. I would like to thank my department - Prof. S. Arun Kumar for his invaluable anecdotes and helping me build strong foundations in Theoretical Computer Science, Prof. Sanjeeva Prasad - for his impeccable style in teaching Programming Languages and Logic to Computer Science, Prof. Shyam Kumar Gupta for his trust and confidence in me, honest criticism, and a source of mysteries of Mathematics.

I would also like to thank IMPECS and Microsoft Research for conducting Workshops on Algorithms at IIT Delhi and Bangalore. To my Algorithm Reading Group friends - Syamantak, Manoj, Ankur, Divya, Nishita, Deepak,

Ankit, Sahil, Arpit, Arindam, Prof. Ragesh Jaiswal, Prof. Amit Kumar, Prof. Naveen Garg, Prof. Sandeep Sen, for giving talks on exciting topics and expanding the pool of ideas I was exposed to.

Besides, I would like to express my love and many thanks to my friends - Vishal Narula - for patiently listening to me blabber about TSP, Divya - for letting me know when Prof. Naveen Garg could be found in his office and when their project meetings were scheduled, Pragun - for supporting me in my theoretical work even though he could not digest a lot of Mathematics, my mother - for making my project famous as the problems of salesmen in my house, my father my family - who constantly thought my project was related to travel agencies, my grandfather who being an engineer himself asked me a million times about the practical uses of TSP, my sister - who had to be content with the small graphs I drew for her and above all, the infinite fun I had in explaining all of it to them.

Lastly, I would like to thank Aman Gupta, for letting me use the layout of his thesis and structure of his certificates, and to Prof. David Williamson, for setting standards of how a master's thesis should be written.

Swati Gupta
Computer Science and Engineering
Indian Institute of Technology, Delhi

Abstract

In this project, we consider a restriction of the Traveling Salesman Problem which is formally stated as - 'Given the costs associated with traveling between any pair of n cities, find the tour of the minimum cost which visits each city once and exactly once '. Held and Karp [4] formulated a lower bound for TSP using 1-trees in 1970. The value of this lower is equal to the value of SubTour LP and is conjectured to have an integrality gap of 3/4. Motivated by obtaining a 4/3 approximation for the traveling salesman problem using the Held-Karp bound, we consider the special case when distances satisfy the graph metric on an underlying unweighted graph G. When G is 2-vertex-connected and has a Hamiltonian path, we show how to obtain a spanning Eulerian trail of length atmost (4/3)n. When G is 3-regular 3-edge-connected, the Held-Karp bound is n and we show a novel approach of finding a 4/3 approximation for TSP on G.

During the course of the project, we also looked at properties of graphs that are LP-oblivious and at the structure of half-integer vertices. We give comparisons of our work to recent unpublished results, and detail further directions for research.

Contents

List of Figures

Chapter 1

Introduction

The Traveling Salesman Problem is one of the oldest and the most extensively worked on problems in the field of algorithms. It is formally stated as : Given the costs associated with traveling between any pair of n cities, find the tour of the minimum cost which visits each city once and exactly once. This is equivalent to finding the minimum cost hamiltonian cycle in a complete weighted graph. Mathematically, suppose we have costs c_{ij} for 1 $\leq$ i, j $\leq$ n, associated with going between each pair of cities, then we want to find a cyclic permutation σ_n such that

$$\sum_{i=1}^{n} c_{i\sigma_n(i)} = min_{\tau cyclic} \sum_{i=1}^{n} c_{i\tau_n(i)}$$

.

TSP is an NP Hard problem that is NP-Hard to approximate. The proof of the same can be found in the book Approximation Algorithms [28]. Some restrictions that make the problem approximable are - metric TSP, assymetric

TSP with triangle inequality and symmetric (1,2) TSP. For symmetric TSP, the distances $c_{ij} = c_{ji}$ for all i,j $\in$ [1,n] and for assymetric there is no such condition. Edges cost in 1,2-TSP are restricted to 1 and 2 only. The best known tour constructing algorithms for these three cases give an approximation factor of - 3/2 (for metric TSP, refer to [3]), log(n)/loglog(n) (for asymmetric with triangle inequality, refer to [21]) and 8/7 (for 1,2 symmetric TSP, refer to [22]). Another interesting restriction to TSP is Graphical TSP where the metric used is *graph metric*. Given an undirected, unweighted graph G = (V,E), the cost between two vertices i and j is given by the length of the shortest path between i and j $\in$ G = (V,E). The solution on this metric, corresponds to a walk in the underlying graph G. Thus, the traveling salesman problem becomes the problem of finding the shortest closed walk on the graph G which visits all vertices of G. The best known algorithm for graphical TSP was 3/2 approximation algorithm given by Christofides in 1978. Two recent unpublished results give a 1.5 - ϵ approximation (Gharan et al [25]) and 1.461 approximation (Svensson et al [23]) for graphical TSP. The goal for Metric TSP is to reach a 4/3 approximation, towards proving the long standing conjecture that there must exist a 4/3 approximation. Working with Graphical TSP that is a subset of Metric TSP simplifies the problem and gives the added advantage of exploiting the structure of the graphs.

The basis of the 4/3 conjecture comes from the Held Karp heuristic where Held and Karp used 1-trees as a relaxation of optimal tours. One way of obtaining a lower bound to an optimization problem is to solve the relaxed problem optimally. In this case, it translates to constructing a tour which

satisfies a subset of the properties of any TSP tour. The relaxed Subtour Linear Program for TSP is given as -

$$minimize \sum_{1 \leq i \leq j \leq n} c_{ij} x_{ij}$$

$$subject\ to: \ x(\delta(v)) = 2 \ \ \forall \ \ v \in V$$

$$x(\delta(S)) \geq 2 \ \ \forall \ \ \phi \subset S \subset V$$

$$x_e \geq 0$$

$$x_e \leq 1$$

It was proven in [4] that the value of Held-Karp hueristic is equal to the optimum value of the Sub-Tour LP. The Held-Karp heuristic typically generates solutions of cost above 99% of the optimal. But, there is a known example involving a subcubic graph where the solution obtained is $3/4\mathcal{OPT}$. Since no example worse than this is known, it is conjectured that the integrality gap of the SubTour LP is 3/4 and there exists a 4/3 approximation algorithm for the TSP.

The objective of this project is to study instances of TSP for which the Held-Karp heuristic gives the optimal solution, and develop algorithms on these instances in support of the conjecture. We have restricted ourselves to the *graph metric*. We consider a simple question of whether a Hamiltonian path in a 2-vertex-connected graph can be converted into a spanning Eulerian trail. For this, we give an algorithm (referred to as the *path* algorithm)to convert a given Hamiltonian path into a spanning Eulerian closed trail using atmost

(4/3)n edges. Next, cubic 3-edge-connected graphs are LP oblivious, that is the linear programming formulation gives the trivial bound for n in such graphs. An assignment which achieves the minimum value of n is obtained by simply assigning $x_e = 2/3$ to each of the edges. Gamarnik et al [18] gave an algorithm for these graphs with an approximation factor of $3/2 - 5/389$. Recently Boyd et al [2] also gave 4/3 approximation for cubic graphs and 7/5 for subcubic graphs. We also, in this thesis, give a 4/3 approximation algorithm for cubic 3-edge-connected graphs, referred to as the *cubic* algorithm.

In the subsequent chapters, we have discussed some related work, the *path* algorithm, the *cubic* algorithm, and future directions of research.

Chapter 2

Literature Review

In the book Combinatorial Optimisation [27], William Cook et al have very beautifully explained the **history and developments** in the travelling salesman problem. People have studied the TSP in many different ways. They tried studying the convex polytope of tours and found inequalities which were facet inducing for these polytopes, like in [19, 20]. There were classes of polytopes with different properties and hence, different inclusion relations between them. These inequalities were formed as a generalisation of those which hold for any tour (giving rise to relaxed tours)- for example - the subtour elimination constraints and comb inequalities (which are disjoint and both facet inducing) and then came clique tree inequalities which generalised subtour and comb inequalities. It is quite interesting how a graph which satisifies subtour elimination constraints, does not satisfy a mixture of subtour and degree constraints(which give the comb inequality). Examples to understand the same can be found in the book or through following papers [19], [20]. There were also studies quanitifying the advantage of different

classes of facet inducing inequalities, refer to [24]. Apart from this, Alexander Shrijver's book on History of Combinatorial Optimization [29] gives an interesting historical account of TSP that dates back to Kirkman and Hamilton.

The main idea behing **N. Christofides' 3/2 approximation algorithm** (1978) for the metric TSP (refer to [3]) is that after ensuring connectivity, one can add a minimum number of edges to get a closed walk. For this, a minimum spanning tree was first found out, then the vertices with odd degree were matched with a matching of cost no more than $1/2\mathcal{OPT}$. This ensured that the degree of all vertices was even, thus giving an Eulerian graph. Hence, there exists a closed walk of cost no more than $3/2\mathcal{OPT}$. A generalisation of this technique is *T-joins*. A *T-join* of G = (V,E) is a set of edges J such that $\mid J \cap \delta(v) \mid \equiv \mid T \cap \{v\} \mid (mod 2), \forall v \in V$. T-joins can always be reduced to paths which are edge-disjoint. It is still not clear though how T-joins can be used to improve the best known approximation factor.

An important approach of approximating a problem is by using **lower bounds**. What is the minimum size of a subgraph of G such that every vertex has a degree greater than or equal to 2 *(D2 bound)*? Or what is the smallest 2-edge-connected graph of a subgraph? These questions give a lower bound for the length of any closed tour that visits all the vertices of the graph. In their Bachelor's thesis, Kushal et al [9] explain these lower bounds in detail. They give relations between the toughness of graph, the D2 bound, the ear decomposition bound and introduce a new bound - *Durability bound*. Another important lower bound is that developed by **Held and Karp** in

[4]. The Held-Karp hueristic uses 1-trees that are minimum spanning trees with an extra edge incident on vertex 1 that makes a loop. Using lagrangian multipliers, they make sure that the degree of all nodes was as close to 2 as possible. More analysis on the Held-Karp hueristic can be found in David Williamson's master thesis (refer to [18]). It was in this thesis that the **4/3 conjecture** was made.

We tried to characterize graphs which have the Held-Karp bound equal to n. **SEP feasible graphs** are those where an assignment of positive real values to the edges of the graph satisfies the constraints of the Subtour LP, refer to [17]. Such graphs are necessarily 2 vertex-connected, 1-tough and 1-block-tough. Hamiltonian graphs are also SEP feasible and hence these become necessary conditions for hamiltonianicity. As an aside, a graph that is t-block-tough is also t-tough. Hence, block-toughness becomes a stronger condition than toughness, but not a sufficient condition.

Further we explored **edge-toughness** introduced by Katona et al in [7] and [12]. The necessary condition for Hamiltonian graphs is that the size of a cut set S of vertices has to be greater than the number of components of G\S. For edge-toughness, Katona et al generalize the cut set to a set of vertices and edges. They present it as a tool to prove non-hamiltonicity. Another known concept at that time was of non-path-toughness, for which there was no easy way to prove it for a graph. Non-path-toughness takes a set of vertices (X) and counts the minimum number of disjoint paths required to connect the vertices of X. They prove that t-edge-tough is also t-tough; a hamiltonian

graph is always 1-edge-tough and 2t-toughness implies t-edge-toughness. Katona et al also prove that there exist $(2t - \epsilon)$-tough graphs which are not t-edge-tough. They prove that every 1-edge-tough graph has a 2-factor.

Motivated by some constructions for graphs with held-karp bound equal to n, we looked at **Half-Integral** solutions to the Subtour LP. Robert Carr et al in [6] give a 4/3 tour constructing algorithm for half-integer *triangle*-vertices. Their algorithm considers a very small portion of actual solutions for which the Held-Karp bound is n. Although in this paper, they form an interesting notion of using *patterns*, that we use later in our *path* algorithm.
The following section presents some small observations during the course of the background reading.

Observations

1. Necessary condition for $\|T - join\| \leq n/2$ is that the graph should be 1-factorable.

2. Held-Karp bound for k-regular and k-edge connected graphs is n. This can be seen by simply assigning $x_e = 2/k$ for each edge.

3. Though SEP feasible graphs are 2-vertex connected, 1-tough and 1-block-tough, they are not 1-edge-tough and path-tough. A simple example for an SEP feasible graph which is not 1-edge-tough and path-tough is shown in figure [**??**]. Consider the set of vertices A = v1,v2,v3 and the set of edges Y = a,b,c,d,e,f. Let X = {}. Then (X,Y) acts as an A-separator. Also, considering the same (X,Y), we get the inequality

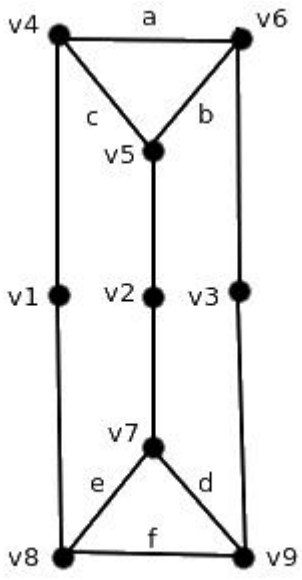

Figure 2.1: An SEP Feasible Graph, that is not 1-edge-tough

for edge-toughness. Thus, proving that this graph is not 1-edge-tough and path-tough. It is SEP-feasible as we can simply assign 1/2 to the edges a,b,c,d,e,f and 1 to others. This hints at the idea that the Held-Karp formulation does not really penalise for disjoint paths, but preserves vertex-related properties(toughness) and connectivity.

4. The construction of graphs in [8] used for disproving 2-tough conjecture has Held-Karp bound = n. Consider any $G(L, u, v, l, 2l + 3)$, refer to [8] for definitions. Charge every instance of L with two 1/2-triangles at vertex u and v and 1-edges otherwise, as shown in the figure [2.2]. Next, since each occurence of u and v is connected, join these through 1

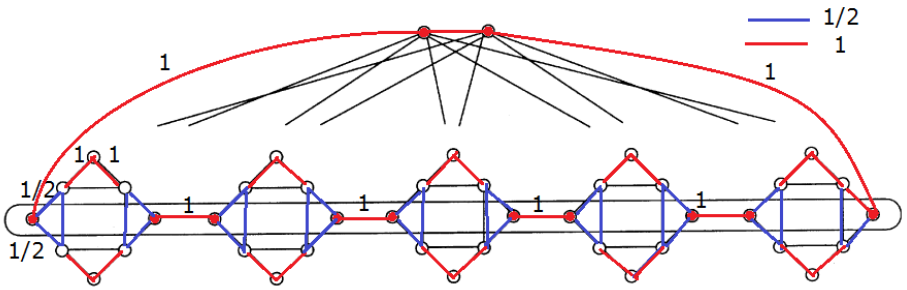

Figure 2.2: Counter-Example

edges to form a path of the L graphs. Since K_l again has a Hamiltonian path between any two vertices, raise the edges of the path to 1 to get a satisfying x_e assignment. An example with G(L,u,v,2,5) is shown in the figure [2.2].

We started with a simple question - Given a Hamiltonian Path in a 2-vertex connected graph, can we find a walk of length at most 4/3(n) covering all the vertices? We found an interesting partition of edges which could be used to solve this problem. This algorithm is referred to as the *path* algorithm in this thesis, and detailed in Chapter 3.

In the paper by Gamarnik et al [5] give a 3/2-5/389 approximation algorithm for cubic 3-edge-connected graphs, as a step towards supporting the 4/3 conjecture. Bill Jackson et al in [13] prove that for graphs with 3-edge-connectivity, there exists a spanning even sub-graph such that the size of each component is atleast 5. To prove this result, they first prove a stronger statement that for a 3-edge-connected graph G such that there exists a vertex u with degree = 3 and two edges u_1 and u_2 incident on u, there

exists a spanning even subgraph X with $\{e_1, e_2\} \subset E(\mathrm{X})$ and $\sigma(X) \geq 5$. We convert their proof into an algorithm for finding an even-spanning subgraph for a given graph G that satisfies these properties. Using this, we give a 4/3 approximation for cubic 3-edge-connected graphs. This algorithm, called the *cubic* algorithm in this thesis, is detailed in Chapter 4.

Chapter 3

Tours using Hamiltonian paths

Problem Statement: *To find a tour of length (4/3)n, given a hamiltonian path in an undirected hamiltonian graph.*

3.1 Definitions

Let P be the hamiltonian path and let the vertices on the path be labelled as $\{v_1, \ldots, v_n\}$, where n is the number of vertices in the graph. We will follow the convention that for every edge (v_s, v_t), $s < t$. Any edge (v_s, v_t) is said to be *l-incident* at v_s and *h-incident* at v_t. From the set of edges $\{(v_s, v_k)$: k $\in$ I$\}$ incident on a vertex v_s, its *deepest-edge* is (v_s, v_t) such that $t >= k \ \forall$ k $\in$ I.

Deepest-edge in an interval of vertices is the one which is h-incident on the highest indexed vertex and l-incident on a vertex in the interval. Two

deepest-edges (v_s, v_t) and (v_k, v_l) are said to be *d-adjacent* if $s < k < t < l$(or $k < s < l < t$).

3.2 Algorithm

1. **Building a set of deep edges**

   (a) Include the deepest-edge from the vertex v_1 in S.

   (b) If the last added deepest-edge is (v_k, v_l), include in S the deepest-edge in the interval $[v_1, v_{l-1}]$.

   (c) Repeat Step 1b till an edge h-incident on v_n is included.

   (d) Label the edges in the order of addition as $\{e_1, \ldots, e_K\}$.

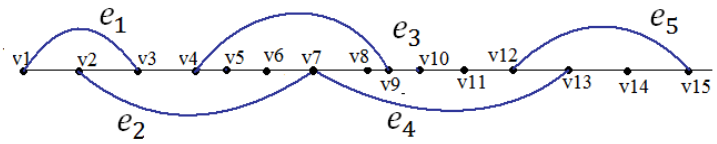

Figure 3.1: Deepest-edges

Claim 1: Step 1 will terminate.

Suppose there is a vertex v_i, such that the deepest-edge in the interval $[v_1, v_{i-1}]$ does not cross the vertex v_i then v_i becomes a cut-vertex and violates the hamiltonianicity of the given graph.

Claim 2: For $i > 2$, the deepest-edge e_i is l-incident on $[v_t, v_{l-1}]$ where e_{i-2}=(v_s, v_t) and e_{i-1}=(v_k, v_l). If it was not so, then either the fact that it is deepest in the interval $[v_1, v_{l-1}]$ will be violated or

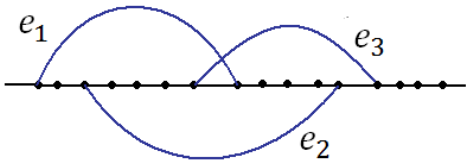

Figure 3.2: Deepest edges will never cross each other

the selection of an earlier edge would be proven to be false. That is, a situation like Figure 2 can never arise. Only the 'deeper' of these two edges(which is e_3 in this case) would have been picked in Step 1, instead of e_2.

2. **Splitting vertices**: Split every vertex v_k on which two deepest-edges are incident into 2 vertices v_{k1} and v_{k2} such that the deepest-edge which is h-incident on v_k becomes h-incident on v_{k1} and the other on v_{k2}. The example in Figure 1 thus becomes as shown in Figure 3. Note that edges which were d-adjacent in the previous graph remain d-adjacent in the new graph as well. Also, two deepest-edges which are incident on the same vertex are never d-adjacent.

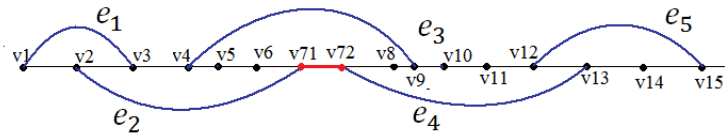

Figure 3.3: Modified Graph

3. **Defining Intervals**: Now, define the intervals formed by the end-points of deepest-edges in the set S as $\{x_i\}$. For K deepest edges, the

number of intervals formed will be exactly 2K-1. The intervals for the modified graph in Figure 3 thus become as shown in Figure 4.

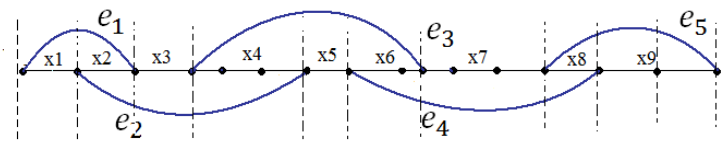

Figure 3.4: Intervals defined on the modified graph

4. **Forming Patterns**: Define three patterns P_i for i={1,2,3} using intervals $\{x_j : j \equiv i \ mod \ 3\}$ as follows -

   $P_1 : x_1 \ e_2 \ x_4 \ e_3 \ x_7 \ e_5$

   $P_2 : e_1 \ x_2 \ e_2 \ x_5 \ e_4 \ x_8 \ e_5$

   $P_3 : e_1 \ x_3 \ e_3 \ x_6 \ e_4 \ x_9$

   These three patterns for the modified graph are shown in the Figure 5. Note that every pattern takes pairs of d-adjacent deepest-edges separated by some intervals x_i, which we will henceforth refer to as the connecting intervals.

5. **Selecting a pattern**: Evaluate the cost of patterns P_i by adding 1 for each deepest-edge used in the pattern and adding the number of vertices in the each interval x_j used in it. The patterns formed above have a cost of 7, 4 and 5 respectively. In general, the total cost of the patterns, C is -

   $$C = \sum x_i + 2K$$

   since each interval is used exactly once and each deepest-edge is used

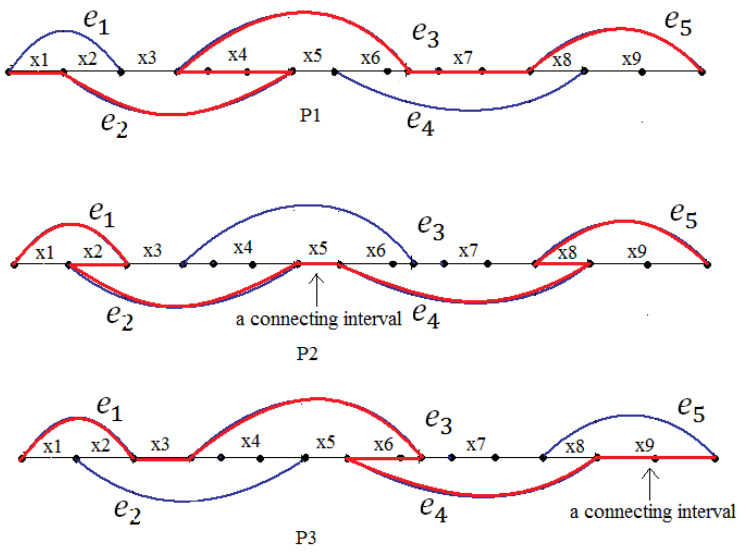

Figure 3.5: Patterns formed for the modified graph

twice. But

$$\sum x_i = n - 2K$$

as vertices in the intervals and the end points of the deepest-edges add up to the total number of vertices. Hence, the total cost of the patterns = n. Hence, atleast one of the patterns must have a cost <= $\lfloor n/3 \rfloor$. In our example, this pattern is P_2.

6. **Forming the walk**: To finally form the walk, we add the deepest-edges in the pattern as it is. For every interval, edges equal to the number of vertices in each interval are added to connect the vertices to the deepest-edge connecting that interval in the pattern(can be any one of the deepest-edges). Refer to Figure 6 for the two options. In

case an interval did not contain any vertices, no edge is added to it.

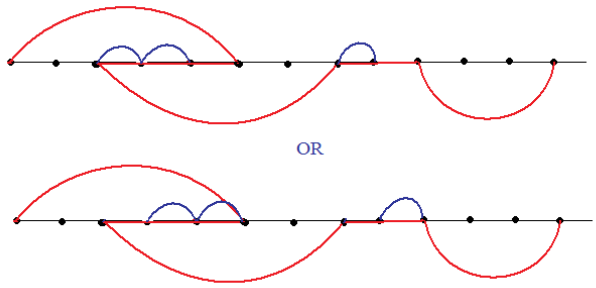

Figure 3.6: Connecting vertices in the intervals in 2 ways

Doing this gives us intervals which are basically Eulerian, except an edge which makes the last vertex in the interval and another vertex of the other deepest-edge as odd degree vertices. Removing this edge from the interval between a d-adjacent pair of deepest-edges does not disconnect the graph. But removing this edge from the connecting interval between two d-adjacent pairs disconnects the graph, as shown in Figure 8. Thus, we will remove the extra edges from the interval enclosed in the d-adjacent pair, as shown in Figure 7, and use these to double the edges in the connecting intervals to make the graph Eulerian, as shown in Figure 9.

Now, every pattern has atleast $\lfloor K/3 \rfloor$ pairs of d-adjacent deep-edges. One edge from each such pair can be removed without disturbing the connectivity of the graph. Thus giving us atleast $\lfloor K/3 \rfloor$ extra edges. Also, note that each vertex which was split in the earlier step had

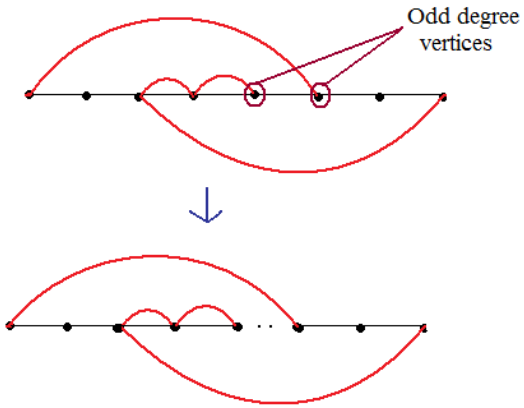

Figure 3.7: Intervals inside d-adjacent deepest-edges

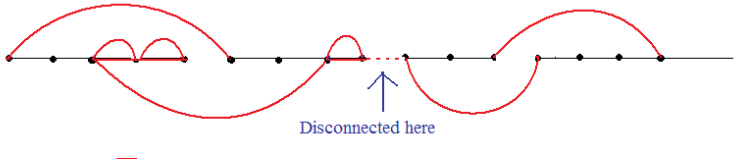

Figure 3.8: Removing an edge from connecting intervals disconnects the graph

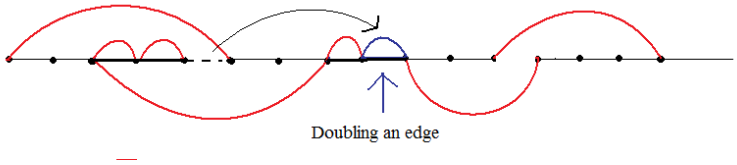

Figure 3.9: Doubling an edge in the connecting interval

to define the boundaries of a connecting interval(interval between a pair of d-adjacent deepest-edges). Thus, combining these vertices back does not disconnect the graph or increase the number of edges of used. After combining back the split vertices if any, there are atmost $\lceil K/3 \rceil$ connecting intervals. Hence, to make the graph Eulerian we will remove atmost 2 more edges. Thus, we get a walk using not more than $\lceil 4/3 \rceil n$ edges. Final walk for the example we were working on is shown in Figure 10.

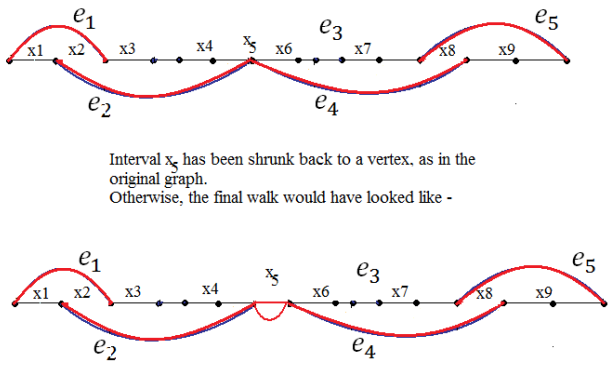

Figure 3.10: Final walk formed using P_2

Chapter 4

Cubic 3-Edge-Connected Graphs

We consider the special case of the problem when G is 3-regular (also called cubic) and 3-edge-connected. Note that the smallest Eulerian subgraph contains at least n = $\|V\|$ edges. In fact, in the shortest path metric arising out of such a graph the Held-Karp bound for the length of the TSP tour would also be n. This is because we can obtain a fractional solution to the sub-tour elimination LP (which is equivalent to the Held-Karp bound) of value n by assigning 2/3 to every edge in G.

Improving the approximation ratio for metric-TSP beyond 3/2 is a long standing open problem. For the metric completion of cubic 3-edge connected graphs Gamarnik et.al. [18] obtained an algorithm with an approximation guarantee slightly better than 3/2. The main result of this paper is to improve this approximation guarantee to 4/3 by giving a polynomial time

algorithm to find a connected Eulerian subgraph with at most 4n/3 edges. This matches the conjectured integrality gap for the sub-tour elimination LP for the special case of these metrics.

Problem Statement: *Given a cubic 3-edge-connected graph G, give a tour that spans all vertices and uses atmost* $(4/3)n$ *edges.*

4.1 Preliminaries

Let $N_G(x)$ denote the neighbours of x and let $d_G(x)$ denote the degree of x. Let $E(v)$ be the set of edges incident on v. For a given subgraph H of G, we can contract every edge in H to a vertex. The resulting graph is represented as G/H, and the vertex in G/H corresponding to H is called a super vertex represented as [H]. σ(X) is used to represent the minimum size of components of X.

4.2 Important Lemmas

Lemma 1 [15]: *Every bridgeless cubic graph has a 2-factor.*

Lemma 2 [14]: *Let G be a k-edge-connected graph,* $v \in V($G$)$ *with d(v)* $\geq$ *k+2. Then there exists edges* e_1,$e_2 \in E($G$)$ *such that* $G_v{}^{e_1,e_2}$ *is homeomorphic to a k-edge-connected graph.*

Lemma 3 [13]: *Let G be a 3-edge-connected graph with n vertices. Then G has a spanning even subgraph in which each component has atleast min*$\{5,n\}$ *vertices.*

4.3 Algorithm

Our algorithm can be broadly split into three parts. We first find a 2-factor of the cubic graph that has no 3-cycles and 4-cycles. Next, we compress the 5-cycles into 'super-vertices'and split them using Lemma 2 to get a cubic 3-edge-connected graph G_0 again. Repeatedly applying the first part on G_0 and compressing the five cycles gives a 2-factor with no 5-cycle on the vertices of the original graph. We 'expand'back the super-vertices to form X that is a subgraph of G. We finally argue that X can be modified to get a connected spanning even multi-graph using at most 4/3(n) edges.

The starting point of our algorithm is Theorem 3 [13]. In fact [13] proves the following stronger theorem.

Theorem: *Let G be a 3-edge-connected graph with n vertices, u_2 be a vertex of G with d(u_2) = 3, and $e_1 = (u_1, u_2), e_2 = (u_2, u_3)$ be edges of G. (It may be the case that u1 = u3). Then G has a spanning even subgraph X with $\{e_1, e_2\} \subset E(X)$ and $\sigma(X) \geq$ min(n,5).*

The proof of this theorem is non-constructive. We refer to the edges e_1, e_2 in the statement of the theorem as 'required edges'. We now discuss the changes required in the proof given in [13] to obtain a polynomial time algorithm which gives the subgraph X with the properties as specified in the Theorem above. Note that we will be working with a 3-regular graph (as against an arbitrary graph of minimum degree 3 in [13]) and hence the even subgraph X we obtain will be a 2-factor.

1. If G contains a non-essential 3-edge cut then we proceed as in the proof of Claim 2 in [13]. This involves splitting G into 2 graphs G_1,G_2 and

suitably defining the required edges for these 2 instances so that the even subgraphs computed in these 2 graphs can be combined. This step is to be performed whenever the graph under consideration has an essential 3-edge cut.

2. Since G is 3-regular we do not require the argument of Claim 6.

3. Since G has no essential 3-edge cut and is 3-regular, a 3-cycle in G implies that G is K_4. In this case we can find a spanning even subgraph containing any 2 required edges.

4. The process of eliminating 4-cycles in the graph involves a sequence of graph transformations. The transformations are as specified in [13] but the order in which the 4-cycles are considered depends on the number of required edges in the cycle. We first consider all such cycles which do not have any required edges, then cycles with 2 required edges and finally cycles which have one required edge. Since with each transformation the number of edges and vertices in the graph reduces we would eventually terminate with a graph, say G_0, with girth 5. We find a 2-factor in G_0, say X_0 and undo the transformations (as specified in [13]) in the reverse order in which they were done to obtain a 2-factor X in the original graph G which has the properties of Theorem 1.

Suppose the 2-factor obtained X contains a 5-cycle C. We compress the vertices of C into a single vertex, say v_C, and remove self loops. v_C has degree 5 and we call this vertex a super-vertex. We now use Lemma 2 to

replace two edges (x_1, v_C) and (x_2, v_C) incident at v_C with the edge (x_1, x_2) while preserving 3-edge connectivity. The edge (x_1, x_2) is called a super-edge. Since the graph obtained is cubic and 3-edge connected we can once again find a 2-factor, each of whose cycles has length at least 5. If there is a 5-cycle which does not contain any super-vertex or super-edge we compress it and repeat the above process. We continue doing this till we obtain a 2-factor, say X, each of whose cycles is either of length at least 6 or contains a super-vertex or a super-edge.

In the 2-factor X we replace every super-edge with the corresponding edges. For instance the super-edge (x_1, x_2) would get replaced by edges (x_1, v_C) and (x_2, v_C) where v_C is a super-vertex obtained by collapsing the vertices of a cycle C. After this process X is no more a 2-factor but an even subgraph. However, the only vertices which have degree more than 2 are the super-vertices and they can have a maximum degree 4.

Let X denote this even subgraph.

Consider some connected component W of X. We will show how to expand the super-vertices in W into 5-cycles to form an Eulerian subgraph with at most $4|W|/3 - 2$ edges, where $|W_0|$ is number of vertices in the expanded component. For each component we will use 2 more edges to connect this component to the other components to obtain a connected Eulerian subgraph with at most $4|W|/3$ edges. Note that the subgraph we obtain may use an edge of the original graph at most twice. We now consider two cases depending on whether W contains a super-vertex.

Case i. W has no super-vertices. Then, W is essentially a cycle with atleast 6 vertices. W is already Eulerian and we can use $\lfloor |W|/3 \rfloor$ edges to connect

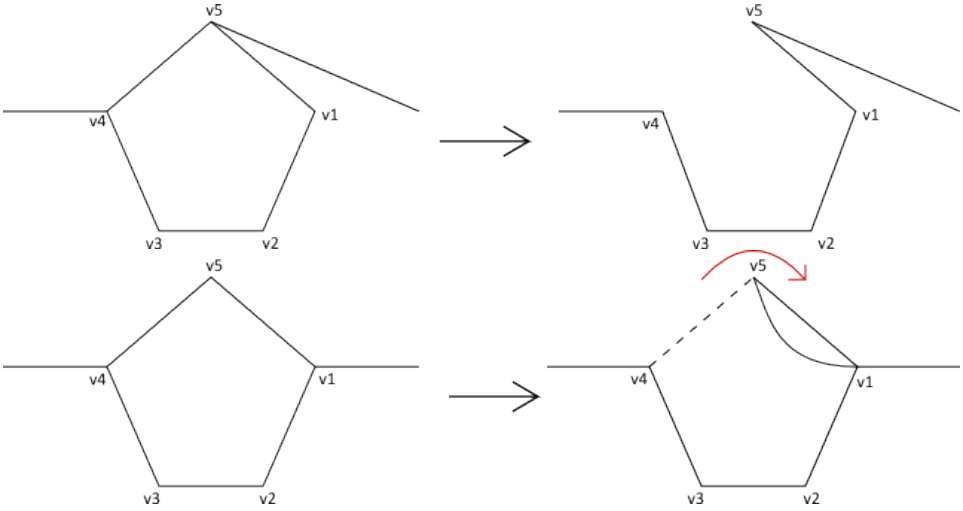

Figure 4.1: On Expanding a super-vertex with degree 2

W to the rest of the graph by doubling an edge ∈ E(G) from a vertex in W to $G \setminus W$.

Case ii. W has atleast one super-vertex, say s. We will discuss the transformations for a single super-vertex. For other super-vertices in the component similar constructions can be done. Now, s can either have degree = 2 or 4. If s has degree 2, then the 2 edges incident on the 5-cycle corresponding to s would be as in Figure [**??**]. In both cases we obtain an Eulerian subgraph. By this transformation we have added 4 vertices and at most 5 edges to the subgraph W.

Suppose the super-vertex s has degree 4 in the component W. W may not necessarily be a component of the subgraph X') as it might have been obtained after expanding a few super-vertices, but that will not effect our algorithm. Let C be the 5-cycle corresponding to this super-vertex and let

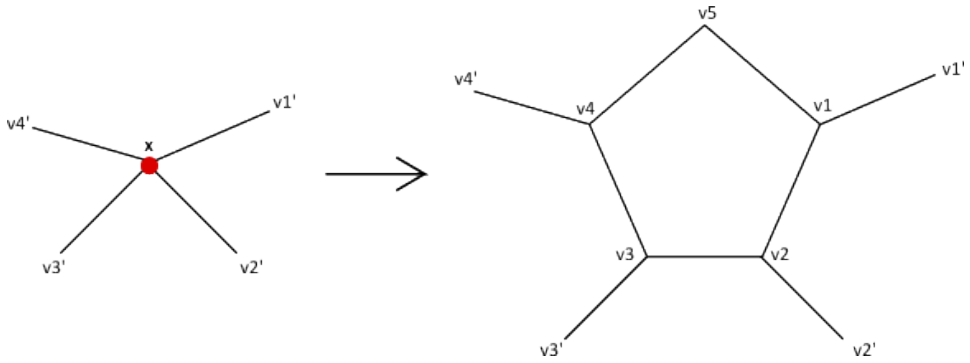

Figure 4.2: On Expanding a super-vertex with degree 4

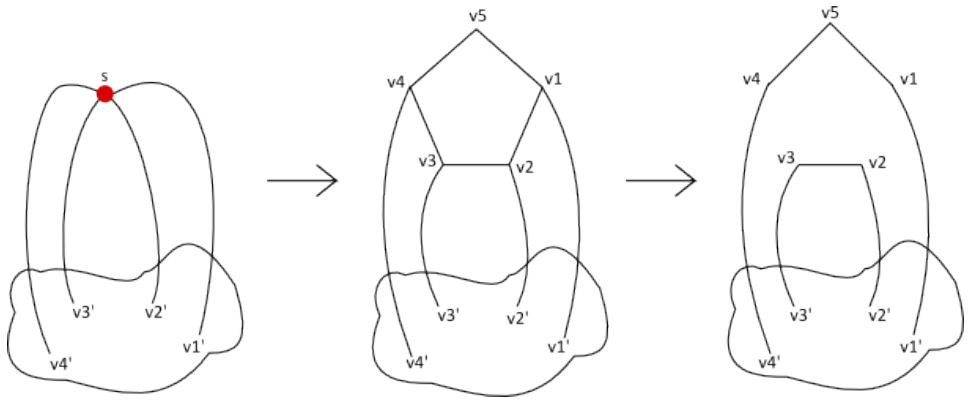

Figure 4.3: Expanding a super-vertex with degree 4

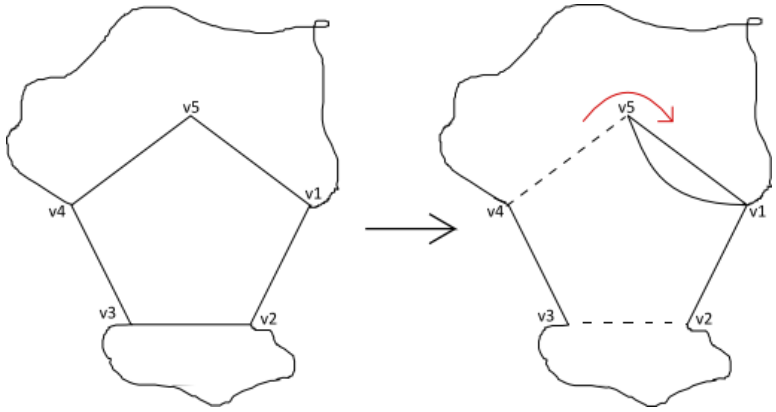

Figure 4.4: When v1,v4 are in the same component

v_1, v_2, v_3, v_4, v_5 be the vertices on C (in order). Further let v_i' be the vertex not in C adjacent to v_i. Let (v_5, v_5') be the edge incident on C that is not in the subgraph W.

We replace the vertex s in W with the cycle C and let W_0 be the resulting subgraph. Note that by dropping edges $\{v_1, v_2\}$ and $\{v_3, v_4\}$ from W_0 we obtain an Eulerian subgraph which includes all vertices of C. However, this subgraph may not be connected as it could be the case that edges $\{v_1, v_2\}$ and $\{v_3, v_4\}$ form an edge-cut in W_0. If this is the case then we apply the transformation as shown in Figure [**??**]. This ensures that W_0 remains connected and is Eulerian. Note that as a result of this step we have added 4 vertices and at most 4 edges to the subgraph W.

We will now prove that the number of edges used to expand W into an Eulerian graph is no more than $\lfloor(|W|/3)\rfloor - 2$.

Analysis Let the component W have k_1 super-vertices of degree 2, k_2 super-vertices of degree 4 and k_3 vertices of degree 2. This implies W has $k_1 + 2k_2 + k_3$ edges. On expanding a super-vertex of degree 2, we add 5 edges in the worst case. On expanding a super-vertex of degree 4, we add 4 edges in the worst case. So, the total number of edges is $N_e = \leq 6k_1 + 6k_2 + k_3$. The total number of vertices in the expanded component is $N_v = 5k_1 + 5k_2 + k_3$. Now, $N_e/N_v = 1 + \frac{k_1+k_2}{5k_1+5k_2+k_3}$. Using the fact that $k_1 + k_2 + k_3 \geq 5$ and that $k_1 + k_2 \geq 1$, we can show that the expanded component uses atmost $\lfloor 4|W'|/3 \rfloor - 2$ edges.

Chapter 5

Future Work

In the previous chapters, two algorithms have been detailed - the *path* algorithm and the *cubic* algorithm. The algorithm discussed in chapter 4 was submitted to arXiv on 18th January 2011. Two recent unpublished results give a 1.5 - ϵ approximation (Gharan et al [25]) and 1.461 approximation (Svensson et al [23]) for graphical TSP. Another paper by Sylvia Boyd et al [2] gives a 4/3 approximation for Cubic Graphs and 7/5 approximation for Sub-Cubic Graphs.

It is tempting to extend the *cubic* algorithm for graphs with higher degree vertices especially since the result in [13] holds for all 3-edge connected graphs. The example of a $K_{3,n}$ demonstrates that this conjecture would be false. A $K_{3,n}$ is 3-edge connected and any connected Eulerian subgraph contains at least 2n edges.

The *path* algorithm and [23] have certain uncanny similarities. They both partition the edges of the depth first tree, though the path algorithm has more control over which edges are picked. Achieving a 4/3 algorithm requires

a lot more control over the structure of matchings and extending the *path* algorithm looks promising.

Lastly, there is still a lot of work to be done to associate the graph properties with the held-karp bound. To achieve a 4/3-approximation, one might think of using the vertices of the SubTour LP as a starting point. Consider the case of cubic 3-edge-connected graphs. These graphs can be decomposed into different sets of matchings and a cycle-cover. Raising each set of matching to 1 and the cycle cover edges to 1/2 gives a feasible solution to the SubTour LP. But it is still not clear how this might help. Exploiting or studying graph properties might be an alternate route towards achieving a 4/3 approximation.

References

1. Francisco Barahona. Fractional packing of T-joins. SIAM Journal on Discrete Mathematics, 17:661 (669), 2004.

2. Sylvia Boyd, Rene Sitters, Suzanne van der Ster, and Leen Stougie. TSP on cubic and subcubic graphs. In Proc. of the 15th Conference on Integer Programming and Combinatorial Optimization (IPCO 2011). To appear.

3. Nicos Christodes. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report 388, Graduate School of Industrial administration, Carnegie-Mellon University, 1976.

4. Michael Held and Richard M. Karp. The traveling-salesman problem and minimum spanning trees. Operations Research, 18:1138(1162), 1970.

5. David Gamarnik, Moshe Lewenstein, and Maxim Sviridenko. An improved upper bound for the TSP in cubic 3-edge-connected graphs. Operations Research Letters, 33(5):467 (474), 2005.

6. S. Boyd and R. Carr, Finding low cost TSP and 2-matching solutions using certain half integer subtour vertices, Report TR-96-12, Department of Computer Science, University of Ottawa, Ottawa, 1996.

7. Gyula Y. Katona, Toughness and Edge-Toughness, Discrete Mathematics, 164, 187 (196), 1997.

8. D. Bauer, H.J. Broersma, H.J. Veldman, Not Every 2-tough graph is hamiltonian, Discrete Applied Mathematics, 99, 317 (321), 2000.

9. Akash M. Kushal, Mohit Singh, Vineet goyal, An Approximation Algorithm for 2-Edge-Connectivity Problem, Bachelors Thesis, Indian Institute of Technology, 2003.

10. Tomas Kaiser, Disjoint T-paths in tough graphs, Journal of Graph Theory, 57, 1097 (0118), 2008.

11. M. Grtschel, M.W. Padberg On the symmetric travelling salesman problem Inequalities, Mathematical Programming (1979) 265-280.

12. G. Y. Katona, Properties of edge-tough graphs, Graphs and Combinatorics. 15 (1999) 315325.

13. Jackson, B. and Yoshimoto, K., Spanning even subgraphs of 3-edge-connected graphs . Journal of Graph Theory, 62: 3747, 2009.

14. W. Mader, A reduction method for edge-connectivity in graphs, Ann. Discrete Math. 3(1978), 145-164.

15. J. Peterson, Die Theorie der regularen Graphen, Acta Math. 15 (1891) 193-220.

16. L. Lovász: On some connectivity properties of Eulerian graphs, Acta Math. Hung. 28 (1976),129-138

17. Boyd, S., Elliott-Magwood, P. - Feasibility of the Held-Karp LP relaxation of the TSP, Technical Report TR-2007-07, SITE, University of Ottawa, Ottawa, Canada, 2007.

18. D.B. Shmoys and D.P. Williamson, Analyzing the Held-Karp TSP bound: A monotonicity property with application, Inf. Process. Lett. 35 281 (285), 1990.

19. M. Grtschel On the symmetric travelling salesman problem Solution of a 120-city problem, Mathematical Programming Study 12, 61-77,1980.

20. M. Padberg, G. Rinaldi - Abranch-and-cutalgorithmfor the resolution of large-scale symmetric traveling salesman problems. SIAM Review, 33(1):60100, 1991.

21. A. Asadpour, M.X.Goemans, Aleksander Madry, S. Oveis Gharan,Amin Saberi - An O(log n/ log log n)-approximation Algorithm for the Asymmetric Traveling Salesman Problem. Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, 2010.

22. P. Berman, M. Karpinski - 8/7 Approximation Algorithm for (1,2) TSP. Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, 2006.

23. T. Momke, Ola Svensson - Approximating Graphic TSP by Matchings, 2011 [arXiv].

24. M.X. Goemans, Worst-case Comparison of Valid Inequalities for the TSP, Mathematical Programming, 69, 335-349, 1995.

25. Shayan Oveis Gharan, Amin Saberi and Mohit Singh - A Randomized Rounding Approach to the Traveling Salesman Problem, Preprint, December 2010, Updated April, 2011.

26. N. Aggarwal, Naveen Garg, Swati Gupta - A 4/3 Approximation of TSP on cubic 3-edge-connected graphs, 2011 [arXiv].

27. Combinatorial Optimization. William J. Cook, William H. Cunningham, William R. Pulleyblank, Alexander Schrijver.

28. Approximation Algorithms. Vazirani, Vijay V.

29. Alexander Shrijver, History of Combinatorial Optimization, 2003.