New Jersey Institute of Technology

Digital Commons @ NJIT

Spring 5-31-2005

Approximation algorithms for variants of the traveling salesman problem

Ankur Gupta
New Jersey Institute of Technology

Follow this and additional works at: https://digitalcommons.njit.edu/theses

Part of the Computer Sciences Commons

ABSTRACT

APPROXIMATION ALGORITHMS FOR VARIANTS OF THE TRAVELING SALESMAN PROBLEM

by
Ankur Gupta

The traveling salesman problem, hereafter abbreviated and referred to as TSP, is a very well known NP-optimization problem and is one of the most widely researched problems in computer science. Classical TSP is one of the original $NP-hard$ problems [1]. It is also known to be $NP-hard$ to approximate within any factor and thus there is no approximation algorithm for TSP for general graphs, unless $P = NP$. However, given the added constraint that edges of the graph observe triangle inequality, it has been shown that it is possible achieve a good approximation to the optimal solution [2]. TSP has a number of variants that have been deeply researched over the years. Approximations of varying degrees have been achieved depending on the complexity presented by the problem setup. An obvious variant is that of finding a maximum weight hamiltonian tour, also informally known as the "taxicab ripoff problem". The problem is not equivalent to the minimization problem when the edge weights are non-negative and does allow good approximations. Also important is the problem when the graph is not symmetric. The problem in this case, as should be expected, is slightly tougher to approximate. Another very well researched problem is when weights of edges are drawn from the set $\{1, 2\}$. This study was focused on gaining an understanding of these algorithms keeping in mind the primary endeavor of improving them. This thesis presents approximation algorithms for the aforementioned and other variants of the TSP, and is focused on the techniques and methods used for developing these algorithms.

APPROXIMATION ALGORITHMS FOR VARIANTS OF THE TRAVELING SALESMAN PROBLEM

by
Ankur Gupta

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer Science

May 2005

APPROXIMATION ALGORITHMS FOR VARIANTS OF THE TRAVELING SALESMAN PROBLEM

Ankur Gupta

Dr. Artur Czumaj, Thesis Advisor   Date
Associate Professor of Computer Science,

Dr. David Nassimi, Committee Member   Date
Associate Professor of Computer Science, NJIT

Dr. Alexandros Gerbessiotis, Committee Member   Date
Associate Professor of Computer Science, NJIT

Blank Page

BIOGRAPHICAL SKETCH

Author: Ankur Gupta

Degree: Master of Science

Date: May 2005

Undergraduate and Graduate Education:

- Master of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, USA, 2005

- Bachelor of Science in Computer Science,
  Guru Nanak Dev University, Punjab, India, 1999

Major: Computer Science

I dedicate this thesis to Dr. Artur Czumaj

ACKNOWLEDGMENT

I would like to thank my adviser Dr. Artur Czumaj for the excellent guidance and support during the course of this thesis.

Prof. Czumaj introduced me to the wonderful and challenging field of "Theoretical Computer Science". His knowledge and insight into the field has always been an inspiration to me. Patience, passion, and perseverance are qualities essential to do well in this field and I hope I can inherit some of these qualities from him. I thank Prof. Czumaj for showing me the right track whenever I have wandered and for putting up with my often stupid antics.

I would also like to thank Prof. Gerbessiotis for the excellent courses I took with him and Prof. Nassimi for his support and review of the thesis.

I also want to thank my family (immediate and distant) for supporting me during the course of my study at NJIT.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

THE TRAVELING SALESMAN PROBLEM

Given a graph $G = (V, E, w)$, such that w is a weight function that associates a weight with each edge in E. Classical TSP is the problem of finding the minimum weight hamiltonian tour in the graph. The problem has applications in such diverse areas as logistics, telecommunication networks, circuit board designing, VLSI, genome sequencing, genetic engineering etc. TSP is very well known to be $NP - hard$ and is also $NP - hard$ to approximate and thus there is no approximation for it unless $P = NP$. However given the added constraint that edge weights in the graph observe *triangle inequality* (ie. $w(u, v) \leq w(u, w) + w(v, w) \ \forall \ u, v, w \in E$), there exists a 3/2-approximation for the problem as shown by Christofides [2] in his celebrated paper. This result however has withstood all improvement attempts in the last three decades. There are however a number of other variants of this problem that have received quite some attention in the last couple of years and this has resulted in improved approximations for these problems. This thesis presents the best known approximations for some of these variants. The results are are organized in chapters with each chapter presenting problems that have similar basic constraints with some variations.

Roadmap : Chapter 2 presents the important concept of *cycle cover*. Finding a cycle cover in a graph is an important basic step in a number of algorithms presented in this thesis. A maximum or minimum weight cycle cover gives an upper or lower bound on the weight of an optimal maximum or minimum weight TSP tour respectively. This is an important property, since if starting with a cycle cover, a TSP tour can be obtained with some restriction on the weight added or lost by addition or deletion of edges, then this could give interesting approximation results. As is shown later, a

number of algorithms are doing exactly this and are achieving good approximations by clever additions and deletions of edges. The chapter also presents an important procedure for finding a pair of cycle covers in a graph. The pair of cycle covers has weight guarantee at least or at most twice the optimal maximum or minimum weight TSP tour. This has then been used to achieve some interesting results for a number of TSP variants.

Chapter 3 deals with the problem of finding a maximum weight hamiltonian tour (*MAX-TSP*) in an undirected graph. Two approximation algorithms are presented for MAX-TSP. The first of these is a 3/4-approximation algorithm and the next is an improved randomized algorithm that gives an approximation guarantee of r where $r < 25/33$. Next the same problem is considered for graphs in which the edge weights obey triangle inequality (*metric-MAX-TSP*). The added restriction allows for improved approximations for the problem as is borne out by the 7/8-approximation algorithm.

Chapter 4 deals with the same problem in a directed graph (MAX-ATSP). First a 2/3-approximation algorithm is presented for this problem. The algorithm uses the pair of cycle covers presented in Chapter 2. Next an approximation algorithm for the metric version of the same problem is presented. Here a 10/13-approximation is achieved, again using the pair of cycle covers.

Chapter 5 looks at the minimization problem in undirected graphs. The added constraint is that the edge weights are drawn from the set $\{1, 2\}$. A 7/6-approximation algorithm is presented for *MIN-TSP-1,2*. In Chapter 6, a $0.841 \cdot \log n$-approximation is presented for the minimization problem is directed graphs with triangle inequality (*metric-MIN-TSP*).

Chapter 7 presents the relationship between TSP and connectivity problems. Specifically, it is shown in undirected graphs with edge weights from the set $\{1, 2\}$,

Min \ Max	Symmetric \ Asymmetric	Metric \ Non-Metric	Weights	Approximation
Max	Symmetric	Non-Metric	None	**25/33** [3]
Max	Symmetric	Metric	None	**7/8** [4]
Max	Asymmetric	Non-Metric	None	**2/3** [5]
Max	Asymmetric	Metric	None	**10/13** [5]
Max	Asymmetric	Non-Metric	{1, 2}	**3/4** [6]
Min	Asymmetric	Metric	None	**0.841 · log(n)** [5]

Table 1.1 Summary of Approximation Results.

the minimum weight TSP tour has the same weight as the minimum weight 2-edge-connected subgraph.

The table presents a summary of some of the approximation results presented in this thesis.

CHAPTER 2

CYCLE COVER

2.1 Introduction

Given a graph $G = (V, E, w)$, such that w is a weight function that associates a weight with each edge in E. A *cycle cover* of G is a collection of vertex disjoint cycles covering all vertices of G. A cycle cover is also sometimes referred to as a *binary 2-matching* or a *2-factor* as the degree of every vertex in a cycle cover is 2. A *maximum cycle cover* (*minimum cycle cover* is similarly defined) is a cycle cover with maximum weight over all cycle covers of G.

A maximum (or minimum) cycle cover can be obtained in polynomial time (see for eg. Hartvigsen [7] for a $\mathcal{O}(n^3)$ algorithm). Finding a maximum weight cycle cover is an important basic step in a number of approximation algorithms presented in this thesis. It is easy to see that a maximum cycle cover is an upper bound on the optimal solution for any Maximizing TSP problem. This fact can be used to obtain a trivial 1/2-approximation algorithm for MAX-TSP. First, find a maximum cycle cover in the graph and then delete the cheapest edge from each cycle. Then, arbitrarily patch the resulting paths and it is easy to see that this gives a $1/2 \cdot OPT$ weight bound in the resulting graph.

Presented here is a procedure due to Kaplan et al. [5] for finding a pair of cycle covers in a graph with very useful properties. The procedure is presented for weight maximization properties but the same technique can be used for obtaining the minimization properties.

4

$$Maximize \quad \Sigma_{(u,v) \in K(V)} W_{uv} X_{uv} \quad Subject\ to$$

$$\Sigma_u x_{uv} = 1, \quad \forall u \quad (indegree\ constraint)$$

$$\Sigma_v x_{uv} = 1, \quad \forall u \quad (outdegree\ constraint)$$

$$x_{uv} + x_{vu} \leq 1, \quad \forall u \neq v \quad (two-cycle\ constraint)$$

$$x_{uv} \geq 0, \quad \forall u \neq v \quad (non-negativity\ constraint)$$

Figure 2.1 LP for cycle-cover without two cycles.

2.2 Finding a Pair of Cycle Covers

In [5], Kaplan, Lewenstien, Shafrir, and Sviridenko give an important procedure for find a pair of cycle covers C' and C'' in a complete weighted graph G with the following properties.

1. C' and C'' do not share a 2-cycle, ie. if a 2-cycle is a part of C' then C'' does not contain at least one of the edges from that 2-cycle.

2. The total weight of the two cycle covers is at least twice the weight of optimal TSP tour ie. $\Sigma_{e \in C'} w(e) + \Sigma_{e \in C''} w(e) \geq 2 \cdot OPT$

The procedure is presented.

The Algorithm

Let $G = (V, E, w)$ be a complete directed graph without self loops. Let $n = |V|$, $V = v_1, v_2, v_3 ... v_n$, $E = K(V) = (V \times V) \setminus \{(v, v) : v \in V\}$ and w is a weight function assigning non-negative weights to edges; $w_{u,v} \geq 0 \ \forall (u, v) \in E$.

The first goal in the algorithm is to obtain a cycle-cover of the graph without any 2-cycles. The problem does not have a polynomial time solution and the LP given in the figure is used to obtain a fractional solution. The LP was first used in [8].

Let $\{x^*_{uv}\}_{uv \in K(V)}$ be the optimal solution to the LP. The next step is to round off the fractional solution to an integral one by multiplying all variables by their least common denominator D [1] [2]. The solution so obtained represents a multigraph, let this be $G_0 = (V, E_0, w)$, where $E_0 = \{(Dx^*_{uv} \cdot (u,v) \forall u, v \in K(V)\}$ and total weight of this multigraph is at most D times the optimal solution. Also G_0 is D-regular and has at most $\lfloor D/2 \rfloor$ copies of any 2-cycle, since $x_{uv} + x_{vu} \leq D$ in G_0. Denote by $m_{G_0}(e)$ the number of copies of an edge $e \in G_0$. The next step is to decompose G_0 and obtain two cycle covers C' and C'' in G such that

1. C' and C'' do not share a 2-cycle, ie. if a 2-cycle is a part of C' then C'' does not contain at least one of the edges from that 2-cycle.

2. The total weight of the two cycle covers is at least twice the weight of optimal TSP tour ie. $\Sigma_{e \in C'} w(e) + \Sigma_{e \in C''} w(e) \geq 2 \cdot OPT$

To obtain the cycle covers C' and C'' from G_0, assume that D is even, as otherwise all edges can be multiplied by 2 to make it so. Now there are two cases depending on the value of D. If $D \mod 4 = 0$, then divide G_0 into 2 D-regular multigraphs G' and G'' using the following procedure and then pick the heavier of the two subgraphs and iterate the decomposition process.

First, obtain a D-regular undirected bipartite multigraph B from G_0. This can be achieved by having two nodes v'_i and v''_i in B for each node $v_i \in G_0$ and then for each edge $(v_i, v_j) \in G_0$, add (v'_i, v''_j) to B.

Next use the technique first introduced by Alon [9] to partition B into two D/2-regular multigraphs B_1 and B_2. For each edge $e \in B$ with $m_B(e) \geq 2$ divide evenly it's copies among B_1 and B_2 and delete them from B (If $m_B(e)$ is odd then one copy is saved in B). Next find a Euler cycle in all connected components of B

---

[1]D may be exponential in the graph size but $\log(D)$ is polynomial
[2]All logarithms are to the base 2 unless specified otherwise

and divide the edges alternately among B_1 and B_2. Next obtain graphs G' and G'' from B_1 and B_2 by reversing the above procedure.

Lemma 1 *G_1 and G_2 are $D/2$-regular and contain at most $D/4$ copies of any 2-cycle*

Proof. The first step is to divide all edges with multiplicity greater than 2 in B, evenly among B_1 and B_2. Then from the bipartite graph B, the edges are divided alternately from the Euler tour and this implies that the indegree and outdegree of each vertex is evenly divided among B_1 and B_2 and so both are $D/2$-regular. It is also easy to see that there would not be more that $D/4$ copies of any two cycle in any of these graphs. Any 2-cycle has at most $D/2$ copies in the original graph and this means that there are at most $D/2$ copies of at least one of the edges from the 2-cycle. This edge will have at most $D/4$ copies in either of the two subgraphs. □

Let G_1 be the heavier among the two graphs G' and G''. It is easy to see that $w(G_1) \geq D/2 \cdot OPT$. (Note that if D originally was a power of 2 then this process could be iterated $\log(D)$ times to obtain the required cycle covers.)

Now consider the case when $D \mod 4 = 2$. Then extract, two cycle covers C_1 and C_2 from G_0 such that.

1. C_1 and C_2 don't share a 2-cycle.

2. If an edge has multiplicity $D/2$ in G_0, then it appears in exactly one of the two cycle covers.

To get the required cycle covers, obtain a D-regular bipartite multigraph B from G_0 as described above. A pair of perfect matchings in M_1 and M_2 in B will correspond to the required cycle covers. Here's how to obtain the required perfect matchings.

The technique described by Alon [9] is used to find a perfect matching in a D-regular bipartite graph. Let $m = Dn$ be the number of edges in B. Let t be the

minimum integer such that $m \leq 2^t$. Next, obtain a 2^{t+1}- regular bipartite graph B_1 from B as follows. Replace each edge in B by $\lfloor 2^{t+1}/D \rfloor$ copies of itself and obtain a $D \cdot \lfloor 2^{t+1}/D \rfloor$-regular graph $B_1 = (V_{B_1}, E_{B_1})$. Let $y = 2^{t+1} - D \cdot \lfloor 2^{t+1}/D \rfloor$. Since D is even, therefore $D \cdot \lfloor 2^{t+1}/D \rfloor$ is even and so y is also even. To make B_1 2^{t+1}-regular, find two perfect matchings M and M' in B and add $y/2$ copies of each to B_1.

M and M' are obtained as follows. Define $B' = (V_B, E')$ to be the subgraph of B where $E' = \{(a, b) \in B | m_B(a, b) = D/2\}$. Since B is D-regular, the degree of each node of B' is at most two. Complete B' into a 2-regular multigraph A, (if required, use the edges not contained in B), and obtain M and M' by partitioning A into two perfect matchings. Define the edges in $M - B'$ and $M' - B'$ as *bad edges*. There are at most Dn bad edges in B_1 since $y < D$.

Lemma 2 *If the number of copies of an edge e are $\leq D/2$ in B, then there are $\leq 2^t$ good copies of e in B_1.*

Proof. There are two cases to be considered

1. If the number of copies of e in B are exactly $D/2$, then by construction e appears in exactly one of the matchings M and M'. So the number of instances of (a, b) in B_1 is $D/2 \cdot \lfloor 2^{t+1}/D \rfloor + y/2 = 1/2(D \cdot \lfloor 2^{t+1}/D \rfloor + y) = 2^t$ and none of these are bad edges

2. If the number of copies of e in B are less than $D/2$, then it can be similarly shown that there are less than 2^{t+1} good edges.

□

Next, divide B_1 into two 2^t-regular graphs B' and B'' and try to balance the number of good copies of each edge between the two subgraphs. Denote by B_2 the graph among B' and B'' containing at most half of the bad edges of B_1 and so it has at most $\lfloor Dn/2 \rfloor$ bad edges. Now apply the same algorithm to this graph to obtain a

2^{t-1}-regular graph B_3 that contains at most $\lfloor Dn/4 \rfloor$ bad edges. Repeat this process t times to obtain a 2-regular graph B_{t+1} containing no bad edges. Next the graph B_{t+1} is a cycle-cover and so pick the alternate edges to obtain the required matchings M_1 and M_2.

Lemma 3 *The cycle covers C_1 and C_2 corresponding to the two matchings M_1 and M_2 have the following properties.*

1. *C_1 and C_2 don't share a 2-cycle.*

2. *If an edge has multiplicity $D/2$ in G_0, then it appears in exactly one of the two cycle covers.*

Proof. A 2-cycle appears at most $D/2$ times in G_0 and therefore there are at most $D/2$ copies of one of the edges in the 2-cycle in B and at most 2^t copies in B_1 by the lemma above. Since the good copies are divided as evenly as possible, therefore after t iterations of the algorithm, there shall be at most one copy of this edge in the graph B_{t+1}.

If there are exactly $D/2$ copies of an edge in G_0, then by similar argument it can be proved that there will be exactly one copy left in B_{t+1}. $\square$

Now if $w(C_1) + w(C_2) \geq 2 \cdot OPT$, then C_1 and C_2 give the required cycle covers C' and C'' and the process is terminated.

Otherwise, let $G_1 = G_0 - C_1 - C_2$. G_1 is obviously a $(D-2)$-regular graph, and $w(G_1) \geq$D-2$\cdot OPT$. Also notice that $D-2 \mod 4 = 0$. Next proof is presented that there are at most $\frac{D-2}{2}$ copies of any 2-cycle in G_1.

Lemma 4 *There are at most $\frac{D-2}{2}$ copies of any 2-cycle in G_1.*

Proof. This is so because if there are exactly $D/2$ copies of a 2-cycle in G_0, then at least one copy of the edge that appears exactly $D/2$ times in G_0 is present in either C_1 or C_2. $\square$

Now, if $D \mod 4 = 0$ then the graph is decomposed into two subgraphs and G_1 is the heavier of the two. If $D \mod 4 = 2$, then a $(D-2)$-regular graph G_1 is obtained (assuming cycle covers C_1 and C_2 had weight less that $2 \cdot OPT$) and the decomposition is performed again. It is now easy to see that the process needs to be repeated $\mathcal{O}(\log(D))$ times to obtain the required cycle covers C' and C''.

CHAPTER 3

SYMMETRIC MAX-TSP

3.1 Introduction

Given an undirected graph $G = (V, E, w)$, such that w is a weight function that associates a non-negative weight with each edge in E; MAX-TSP is the problem of finding a maximum weight hamiltonian tour. It should be pointed out that non-negativity of edge weights is required as otherwise the problem is equivalent to MIN-TSP(To observe this; negate all the edge weights and find MIN-TSP and this gives MAX-TSP in the original graph). The problem is known to be $MAX\ SNP-hard$ and thus there exists a constant $\epsilon > 0$ for which there is no ϵ-approximation for the problem. Serdyukov [10] gave the best known deterministic approximation algorithm for this problem with a performance guarantee of 3/4 and no further improvements have been made in two decades. Hassin and Rubinstien [3] gave a randomized algorithm with a performance guarantee of p for any fixed $p < 25/33$.

When the edge weights in the graph observe the *triangle inequality* ie. $w(u, v) + w(v, w) \geq w(u, w)$ $\forall$ $(u, v, w) \in E$, then metric-MAX-TSP is the problem of finding a maximum weight hamiltonian tour in a graph. The problem still remains $MAX\ SNP-hard$. Hassin and Rubinstein [4] gave the best known approximation guarantee of 7/8 using a randomized algorithm. The algorithm is based on the 3/4-approximation for the general case developed by Serdyukov [10] and a 5/6-approximation for metric-MAX-TSP by Serdyukov and Kostochka [11].

3.2 Approximation Algorithms for MAX-TSP

Presented here are approximation algorithms for MAX-TSP. The first algorithm is a 3/4-approximation by Anatoly Serdyukov and is the best know deterministic algorithm for the problem. The next algorithm is a randomized algorithm achieving

1. Find a maximum cycle cover C in G. $C = \{C_1, C_2 ... C_n\}$

2. Find a maximum matching M in G.

3. For each cycle C_i in C.

    Transfer an edge from C_i to M, such that M remains a subtour.

4. Complete C into a tour T_1 and M into a tour T_2

5. Return the heavier of the two tours T_1 and T_2.

Figure 3.1 3/4-approximation for MAX-TSP.

slight improvement on this result by Hassin and Rubinstien. This algorithm is based on ideas developed in [10] and an earlier paper that achieved a 5/7-approximation by Hassin and Rubinstien [12].

3.2.1 3/4-approximation for MAX-TSP

Let $G = (V, E, w)$ be a complete undirected graph without self loops. Let w is a weight function assigning non-negative weights to edges. The algorithm is presented in the figure. It is a simple and elegant algorithm with a simple analysis.

The important step in the algorithm is step 3. All the cycles in the cycle cover C need to be broken by transferring edges from the cycle cover to the matching M also taking care that no cycles are formed in the matching as a result of the transfer. It is not to hard to see that it is always possible to do this. The weight of the cycle cover is an upper bound on OPT and the weight of the matching is an upper bound on $1/2 \cdot OPT$ (Observe that an optimal TSP tour can be divided into two matchings and so a maximum matching is at least the weight of the heavier of the two matchings). Therefore the heavier of the two tours will have weight at least $3/4 \cdot OPT$.

1. Find a maximum cycle cover C in G. $C = \{C_1, C_2...C_n\}$

2. For each cycle C_i in C.

   If $|C_i| \leq \epsilon^{-1}$.

      Compute a Maximum weight hamiltonian path H_i in the subgraph induced be vertices of C_i.

   Else

      Obtain H_i by deleting the least weight edge in C_i.

3. Patch all H_i to obtain tour T_1.

Figure 3.2 (Procedure 1) r-approximation for MAX-TSP, where $r < 25/33$.

3.2.2 r-approximation for MAX-TSP, where $r < 25/33$

This algorithm uses two different procedures. The first procedure produces a tour T_1 and the second procedure produces two tours T_2 and T_3. The best of the three tours gives the desired approximation.

The first procedure is presented in the figure. The procedure uses a parameter ϵ to distinguish between *short cycles* and *long cycles*. For the short cycles, a hamiltonian path is computed in the subgraph induced by the vertices of the cycle and for long cycles a path is obtained by deleting the cheapest edge from the cycle. The tour T_1 is obtained by patching the paths arbitrarily.

The second procedure is a little more involved. The first step as in procedure 1 is to compute a cycle cover C on G, $C = \{C_1, C_2...C_n\}$. Let $E' \subset E$ be the set of all edges $e = (u, v) \in E$ such that u and v belong to different cycles in C. Next compute a maximum matching M' using the edges in E' and a matching M using edges in E. For each cycle $C_i \in C$, construct two disjoint nonempty matchings M_i and M_i' using edges from C_i such that $M_i \cup M$ and $M_i' \cup M$ don't have any cycles and all vertices in C_i are covered by $M_i \cup M_i'$.

Lemma 5 *It is always possible to find the matchings M_i and M_i' with the above required properties.*

Proof. Suppose the matchings have to be constructed from cycle C_i. Let the edges in this cycle be $e_1, e_2 \ldots e_k$. Start with any edge and assign edges alternately to the matchings M_i and M_i'. If the addition of an edge (Say e to M_i) creates a cycle in $M_i \cup M$, then skip this edge and add the next edge in the cycle to M_i. It is easy to see that this edge would never create a cycle in $M_i \cup M$. Therefore two consecutive edges from a cycle are never skipped in this procedure and so all the vertices from the cycle will be covered by the two matchings and also note that each of these matchings would have at least one edge.

There are a couple of cases which need to be handled differently by this procedure. The first is if both edges e_1 and e_k are assigned to the same matching (Say M_i). In this case if edge e_2 was assigned to M_i', then just skip edge e_1. Else if e_2 could not be assigned to M_i', then assign e_1 to M_i'. Another case is if both edges e_1 and e_k were skipped because they could not be assigned to, say M_i and M_i' respectively. Then if e_i could not be assigned to M_i, then e_1 is assigned to M_i'. □

Transfer all the edges from either M_i or M_i' from C_i to M with probability 1/2. Observe that since M_i and M_i' are non-empty therefore all the cycle in the cycle cover C will be broken. Patch M arbitrarily to obtain the second tour T_2.

Let P be the set of paths obtained after transferring edges from the cycles in C. Let $M'' \subset M'$ be the set of edges $e = (u, v) \in M'$ such that u and v have degree 1 in P.

Lemma 6 *For every edge $e \in M'$, the probability that it is in M'' is at least 1/4.*

Proof. It is easy to see now that with probability at least 1/2, one of the edges incident to a vertex in a cycle C_i will be transfered to M, therefore the probability

that a vertex has degree 1 in P is at least $1/2$. For any edge $e = (u,v) \in M'$, it is added to M'' if both u and v have degree 1 in P which has probability at least $1/4$ □

Let P^* be the set of paths and C^* be the set of cycles obtained by $M'' \cup P$. For each cycle $C_i^* \in C^*$, randomly select an edge e from $C_i^* \cap M''$ and delete it from C_i^* and then add C_i^* to P^*. Next, arbitrarily patch P^* to obtain the third tour T_3.

Return the tour with the heaviest weight among $\{T_1, T_2, T_3\}$

Lemma 7 *For each edge $e \in M''$, the probability that it is deleted by the deletion step is at most $1/2$.*

Proof. Each cycle $C_i^* \in C^*$ has at least two edges from M''. At most one of these edges will be deleted. □

Theorem 8 *The heaviest of the three tours ie. $max\{w(T_1), w(T_2), w(T_3)\} \geq \frac{25(1-\epsilon)}{33-32\cdot\epsilon}OPT$ and the algorithm runs in polynomial time.*

Proof. Let T be the optimal TSP tour. Denote by T_{int}, the subgraph containing edges of T whose end vertices are in the same cycle in C. Similarly denote by T_{ext}, the subgraph containing edges with end vertices in different cycles in C. Let $w(T_{int}) = \alpha \cdot w(T) = \alpha OPT$.

In procedure 1 , consider the tour T_1 and a short cycle $C_i \in C$. Since a maximum weight hamiltonian tour is computed on the subgraph induced by the vertices of C_i, therefore the contribution of C_i to T_1 is at least the weight of T_{int} in the graph induced by it's vertices. For each long cycle, the cheapest edge is deleted and so the total loss of weight is at most a factor of ϵ. Therefore $w(T_1) \geq (1 - \epsilon)w(T_{int}) \geq (1 - \epsilon)\alpha OPT$

In the second procedure, consider the process to obtain tour T_2. The weight of the maximum matching M is at least $1/2 \cdot OPT$. Let $\delta \cdot OPT$ be the weight of the edges transfered from C to M. Therefore weight of the tour T_2 is at least $(1/2+\delta)OPT$. Now consider the construction of the tour T_3. The weight of the set of paths P obtained

after breaking the cycles in C is $(1 - \delta)OPT$. Now to calculate the weight added to this, observe that the weight of the matching M' is at least $1/2 \cdot w(T_{ext})$ (since M' is maximum matching over edges connecting different cycles of C). For each edge in M', the probability that it is in M'' is at least $1/4$, therefore $w(M'') \geq 1/8 \cdot w(T_{ext})$. Now the edges from M'' are deleted with probability at most $1/2$, therefore the wight of the remaining edges is at least $1/16 \cdot w(T_{ext}) = 1/16 \cdot (1 - \alpha)OPT$. So weight of $w(T_3) \geq ((1 - \delta) + 1/16(1 - \alpha)) \cdot OPT$.

Now $max\{w(T_1), w(T_2), w(T_3)\} \geq max\{((1 - \epsilon)\alpha), (1/2 + \delta), ((1 - \delta) + 1/16(1 - \alpha))\} \cdot OPT$. The left hand side of the equation minimizes when $\alpha = \frac{25}{33 - 32\epsilon}$. The minimum value of the left hand side then obtained is $\frac{25(1 - \epsilon)}{33 - 32\epsilon} \cdot OPT$.

For any given $r < 25/33$, a value of $\epsilon > 0$ can be obtained to get an r-approximation.

A maximum matching on a graph can be obtained in time $\mathcal{O}(n^3)$. The other time consuming part in the algorithm is the process of obtaining a maximum weight Hamiltonian path in the subgraphs induces by short cycles. Using dynamic programming this can be done in time $\mathcal{O}(n^2 2^{1/\epsilon})$. The total complexity then is $\mathcal{O}(n^2(n + 2^{1/\epsilon}))$. For a fixed $\epsilon > 0$, this is $\mathcal{O}(n^3)$.

$\square$

3.3   7/8-approximation for metric-MAX-TSP

Let $G = (V, E, w)$ be a complete undirected graph without self loops. Let w is a weight function assigning non-negative weights to edges. Additionally, the edge weights observe triangle inequality. As mentioned earlier, the 7/8-approximation for metric-MAX-TSP is a randomized algorithm based on the ideas developed in [10] and [11]. The algorithm in [10] is presented in *Section*; 3.2.1. As was the case in these two algorithms, the first step is to find a maximum weight cycle cover

1. For each cycle C_i in C.

   Pick $e, f \in E \cap C_i$ such that both $M \cup \{e\}$ and $M \cup \{f\}$ are subtours.

   Randomly pick one edge between e and f with probability $1/2$. Call it g.

   $P_i := C_i - g$.

   $M := M \cup g$.

2. Let u_i and v_i be the ends of the path P_i.

3. Give each path P_i a random orientation.

4. Form a tour T_i by adding connecting edges between head of P_i and tail of P_{i+1}.

Figure 3.3 (Procedure 1) 7/8-approximation for metric-MAX-TSP.

1. Let $S :=$ set of end nodes of paths in M.

2. Compute a random perfect matching M_s over S.

3. Let C' be the cycle cover obtained from $M \cup M_s$.

4. Delete an edge from each cycle in C'.

5. Arbitrarily complete C' into a tour T_2.

Figure 3.4 (Procedure 2) 7/8-approximation for metric-MAX-TSP.

$C = \{C_1, C_2, \cdots, C_s\}$ and a maximum matching M. Next the procedure 1 and 2 as presented are applied.

Let T be the the heavier of the two tours obtained from the two procedures. Then the following theorem holds.

Theorem 9 *The expected weight of T, $w(T) \geq (\frac{7}{8} - \mathcal{O}(\frac{1}{\sqrt{n}})) \cdot OPT$*

Proof. First note that $w(C) \geq OPT$ and $w(M) \geq 1/2 \cdot OPT$. Next the weight of the tour obtained in Procedure 1 is analyzed. By triangle inequality.

$$w(u_i, u_{i+1}) + w(v_i, v_{i+1}) + w(u_i, v_{i+1}) + w(v_i, u_{i+1}) \geq 2 \cdot w(u_i, v_i).$$

Now,

$$w(T) = \Sigma_{i=1}^{s} w(P_i) + \frac{1}{4} \Sigma_{i=1}^{s} (w(u_i, u_{i+1}) + w(v_i, v_{i+1}) + w(u_i, v_{i+1}) + w(v_i, u_{i+1}))$$

$$w(T) \geq \Sigma_{i=1}^{s} w(P_i) + \frac{1}{2} \Sigma_{i=1}^{s} w(u_i, v_i)$$

This implies that the patching procedure is able to regain at least half of the weight of the edges deleted from each cycle. Now let α be the weight of the edges that were candidates for deletion. The expected weight of the edges that were actually deleted is $\alpha/2$. But as explained above at least half of this weight can be regained. Therefore the weight of the final tour is given by.

$$w(T_1) \geq (1 - \frac{\alpha}{4}) \cdot w(C) \geq (1 - \frac{\alpha}{4}) \cdot OPT. \tag{3.1}$$

Next consider procedure 2. The expected weight of the edges added to M in procedure 2 is $(\frac{\alpha}{2}) \cdot (w(C))$. Consider a vertex v. If this vertex was incident to two candidates chosen for deletion , then $v \notin S$ and if v was not incident to any candidate then certainly $v \in S$. And if, v was incident to one candidate then $v \in S$ with probability $1/2$.

Let $|S| = k + 1$. For $i \in S$, exactly one edge from $\{(i,j)|j \in S - i\}$ is chosen to M_S. Consider an edge $(i,j) \in E \cap (S \times S)$, the probability that this edge is selected to M_S is $1/k$. If (i,j) is selected, charge it's weight $w(i,j)$ in the following manner: Suppose that i is incident to edges $e', e'' \in C$. If none of these edges was a candidate, charge $w(i,j)/4$ to each of e' and e'' (and nothing to e'). Note that it cannot be that both e' and e'' were candidates since in such a case $i \notin S$. The expected weight charged to an edge $(g,h) \in C$ that was not a candidate is then $(1/k)[\Sigma_{r \in S-g} w(r,g)/4 + \Sigma_{r \in S-h} w(r,h)/4]$.

Now by triangle inequality, $w(r,g) + w(r,h) \geq w(g,h)$ so the above sum is at least $w(g,h)/4$. Therefore $w(M_S \geq w(C)(1-\alpha)/4$ and consequently,

$$w(M \cup M_S) \geq (0.5 + \tfrac{\alpha}{2} + \tfrac{1-\alpha}{4})w(C) = (\tfrac{3+\alpha}{4})opt$$

Finally, the algorithm deletes edges from cycles in $M \cup M_S$. Now the claim is that $|S| \geq n/3$. the reason is that the perfect matching computed in Step 1 has all the n vertices of V with degree 1. Then , one candidate from each cycle of C was added to M. The number of added edges is equal to the number of cycles which is at most $n/3$. Therefore, after the addition of these edges the degrees of at most $2n/3$ vertices became 2, while at least $n/3$ vertices remained with degree 1. The latter vertices are precisely the set S, and this proves that $|S| \geq n/3$. Since M_S is a random matching, the probability that an edge of $M \cup M_S$ is contained in a cycle whose size is smaller than $\sqrt{n}$ is bounded from above by

$$\frac{1}{\frac{n}{3}} + \frac{1}{\frac{n}{3}-1} + \cdots + \frac{1}{\frac{n}{3}-\sqrt{n}} \leq \frac{\sqrt{n}}{\frac{n}{3}-\sqrt{n}} = \mathcal{O}(\frac{1}{\sqrt{n}}).$$

(The j-th term in the left-hand side of this expression bounds the probability that a cycle containing exactly j edges from M_S is created.) Therefore, the expected weight of the edges deleted in this step is $\mathcal{O}(1/\sqrt{n})w(M \cup M_S)$ and

$$w(T_2) \geq (\frac{3+\alpha}{4})(1 - \mathcal{O}(\frac{1}{\sqrt{n}})) \cdot opt. \tag{3.2}$$

Combining (3.1) and (3.2) is it obtained that when $\alpha \leq 1/2$, $w(T_1) \geq (7/8)opt$ and when $\alpha \geq 1/2$, $w(T_2) \geq (7/8)(1 - \mathcal{O}(1/\sqrt{n})) \cdot opt$. Thus,

$$w(T) = max\{w(T_1), w(T_2)\} \geq (\tfrac{7}{8} - \mathcal{O}(\tfrac{1}{\sqrt{n}})) \cdot opt.$$

$\square$

CHAPTER 4

ASYMMETRIC MAX-TSP

4.1 Introduction

Given a directed graph $G = (V, E, w)$, such that w is a weight function that associates a non-negative weight with each edge in E; MAX-ATSP (Asymmetric MAX-TSP) is the problem of finding a maximum weight hamiltonian tour. As is the case with MAX-TSP, the problem is $MAX\ SNP - hard$. The problem is also known to be $APX - hard$ and thus there is no PTAS (polynomial time approximation scheme) for the problem unless $P = NP$. The first approximation algorithm for the problem is due to Fisher, Nemhauser and Wosley [13]. They achieved a performance guarantee of 1/2 using a simple algorithm. The algorithm finds a maximum cycle cover in the graph and then deletes the cheapest edge from each cycle and then arbitrarily patches the resulting paths to obtain a tour. Kosaraju, Park and Stein [14] gave the first non-trivial algorithm achieving a performance guarantee of 38/63. A number of recent improvements have been achieved on this result (see for eg. Blaeser [15] and Lewenstien and Sviridenko [8]). The best known result for this problem is the 2/3-approximation achieved by Kaplan et al. [5].

MAX-ATSP has received considerable attention because of it's applications in solving the $shortest - superstring$ problem. Specifically, Breslauer et al. [16] show that a p-approximation for MAX-ATSP implies a $3.5 - (1.5 \cdot p)$-approximation to the $shortest - superstring$ problem. The $shortest\text{-}superstring$ problem is; given a set of strings $s_1, s_2, s_3....s_n$, find the shortest superstring S that has each s_i as a substring. The problem is $MAX\ SNP - hard$ and has applications in DNA sequencing and data compression.
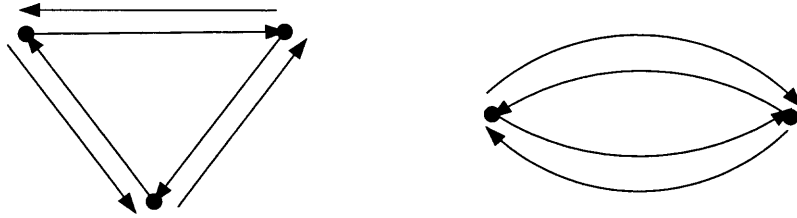
Figure 4.1 Double Oppositely Oriented 2-D-cycle and 3-D-cycle.

4.2 2/3-approximation for MAX-ATSP

In [5] Kaplan, Lewenstien, Shafrir, and Sviridenko give an important procedure for obtaining a 3-path coloring of a 2-regular multigraph. This procedure yields a 2/3-approximation when combined with the procedure presented in section 2.2. The algorithm presented in section 2.2 gives a 2-regular multigraph that has weight at least twice the weight of the optimal TSP tour. It is easy to see that one of the path collections will have weight at least $2/3 \cdot OPT$. The procedure is presented next.

Consider a 2-regular multigraph G that is a combination of two cycles covers C_1 and C_2 such that.

1. C_1 and C_2 do not share a 2-cycle, ie. if a 2-cycle is a part of C_1 then C_2 does not contain at least one of the edges from that 2-cycle.

2. The total weight of the two cycle covers is at least twice the weight of optimal TSP tour ie. $\Sigma_{e \in C_1} w(e) + \Sigma_{e \in C_2} w(e) \geq 2 \cdot OPT$

Each connected component of G will be considered separately. First consider the connected components that are double oppositely oriented cycles. Denote such a pair of cycles of length k as k-D-cycle. Any such cycle is a component by itself.

It is easy to see that there are no 2-D-cycles in G. Consider the 3-D-cycles in G. Such a component cannot be three path colored. But the situation is easily handled by reversing the cheaper cycle in this component and then three path coloring it. Now the process for coloring a 4-D-cycle is given. The same process can be applied

for any k-D-cycle where $k > 3$. Consider a 4-D-cycle over the vertices (v_1, v_2, v_3, v_4). The path coloring is done as follows. The first path collection contains the two edges (v_2, v_1) and (v_3, v_4). The second path collection contains a path from v_1 to v_2 and the third path collection contains a path from v_4 to v_3.

Now consider a connected component of G that is not a D-cycle. Such a component is a combination of two cycle covers. The coloring is done using colors from the set $\{Red, Blue, Green\}$. Color one cycle cover Red and the other one $Blue$. The basic idea now is to break all the Red and Blue cycles in the two cycle covers using $Green$ while ensuring that no Green cycles are formed.

An *alternating path* is defined as a path consisting of alternating red and blue edges. An *alternating cycle* is similarly defined. The algorithm tries to first find a maximal alternating path such that only one edge is picked from any given cycle. Next all the edges on this alternating path are colored green and all the cycles touching this path are removed from the graph. Thus all the cycles on the path are broken. Since the alternating path is maximal, therefore there are no edges in the remaining graph that touch the edges that were colored green and so the edge coloring done after this would not interfere with the edges already colored green. This is important to ensure that no green cycles are formed. The process is repeated till all cycles are broken. The only problem however is that the process to deliver alternating path might deliver an alternating cycle. But an alternating cycles can also be dealt with. Next the process to obtain an alternating path is given.

Start with any edge (v_1, v_2) such that if (v_1, v_2) is in G then it has the same color as (v_1, v_2). Such an edge exists on every cycle C as otherwise there must be a 2-D-cycle which is restricted. Mark such an edge and then extend the path greedily in both directions. For example if (v_1, v_2) is colored red then next the outgoing blue edge from v_2 is marked. Care should be taken when adding new edges to the alternating path, as only edges from cycles that have not been marked can be considered. If the

last edge in the path was colored let say blue and the outgoing red edge incident on the vertex is from a cycle that is already marked, then the process is stopped. Note that suppose there is an alternating path $(v_1, v_2...v_k)$, then an edge is never added from v_k to any vertex from $(v_2, v_3...v_{k-1})$. This is because all these vertices have one red edge incident on them and so adding another red edge is impossible since the red cycle incident on this vertex is already marked. An edge can however be added from v_k to v_1 and that gives an alternating cycle. An alternating cycle is a problem because then this cycle cannot be colored green. But alternating cycles can be handled as demonstrated next.

Consider an alternating cycle $(v_1, v_2...v_k, v_1)$. An alternating cycle always has length ≥ 4. This is so because an alternating cycle always has even length and it will never have length 2 since the algorithm forbids to start with an edge (v_1, v_2) if (v_2, v_1) has a different color. Also note that all edges on the alternating cycle are from different cycles in the two cycle covers.

The situation is handle based on the length of the cycles intersecting the alternating cycle. The first case is when one of the cycles is a 2-cycle. The next is if one of the cycles has length ≥ 4. The last case is when there are at least three consecutive cycles of length 3. It is easy to see that one of these cases must exist.

Consider case 1, and let $(v_i, v_{i+1}), (v_{i+1}, v_i)$ be the edges in the 2-cycle. Assume that the 2-cycle is colored Blue. Then all the edges except (v_i, v_{i+1}) in the alternating cycle are colored Green. The edge (v_i, v_{i+1}) is colored Red. This coloring will not conflict with the other Red edges since both the conflicting Red edges are colored Green. Also this would not form a Red cycle since it joins two disjoint Red paths. Next all the cycles on the original alternating cycle are removed.

Next consider case 2. Let (v_k, v_1) be the edge from the cycle of length ≥ 4 that intersects the alternating cycle. Assume that the cycle is colored Blue. Let (v_1, u) and (u, w) be the next two edges on this cycle. Color Green all the edges on

the alternating path except (v_k, v_1) and remove all the corresponding cycles from the graph. If (u, w) is opposite a Red edge (w, u) then (v_1, u) is colored Red and (w, u) is colored Green. If (u, w) is not opposite a Red edge then a new alternating path search is started using any edge from the cycle except (v_k, v_1). The procedure may end in an alternating path which is good or it may end with an alternating cycle in which case the same process is repeated until either an alternating path is found or an alternating cycle with all cycles of length 3 is found.

Consider case 3, where there are three consecutive cycles of length three. Let the three edges the intersect the alternating cycle be the path (v_k, v_1, v_2, v_3). Next observe that either v_1 or v_2 will not have a 2-cycle incident on it.

4.3   10/13-approximation for metric-MAX-ATSP

Kaplan, Lewenstien, Shafrir, and Sviridenko [5] obtained a 10/13-approximation for metric-MAX-ATSP using a result proved by Serdyukov and Kostochka and the fact they could obtain two cycle covers that weight more than twice the optimal TSP tour and do not share a 2-cycle (as presented earlier).

Given an directed graph $G = (V, E, w)$, where w is a weight function associating a non-negative weight with each edge and the weights obey the triangle inequality. Let C be a maximum weight cycle cover in G. Let W_i be the weight of all the cycles in C containing exactly i vertices. The following lemma is obtained from the result by Serdyukov and Kostochka [11].

Lemma 10 *Given a maximum weight cycle cover C, a hamiltonian cycle H can be obtained in time $\mathcal{O}(n^2)$ such that*

$$\Sigma_{e \in H} w(e) \geq \Sigma_{i=2}^{n} (1 - \frac{1}{2i}) W_i$$

Now let C_1 and C_2 be two cycle covers such that they do not share a 2-cycle and

$$\Sigma_{e \in C_1} w(e) + \Sigma_{e \in C_2} w(e) \geq 2 \cdot OPT$$

Let W_2' be the weight of all the 2-cycles in C_1 and C_2 divided by 2 and let W_3' be the weight of all the other cycles divided by 2. Then the following holds based on the lemma presented above.

$$\Sigma_{e \in H} w(e) \geq \frac{3}{4} W_2' + \frac{5}{6} W_3' \tag{4.1}$$

Now let G' be a graph obtained from the union of all the 2-cycles from C_1 and C_2. Since C_1 and C_2 don't share a 2-cycle therefore G' is a collection of chains consisting of 2-cycles and therefore can be represented as a union of two path collections. If G' contains two oppositely oriented cycles formed by joining 2-cycles , then it can be reversed in the direction of the heavier cycle. Next the two path collections are completed into hamiltonian tours and let H be the heavier of these tours. Then the following holds.

$$\Sigma_{e \in H} w(e) \geq W_2' \tag{4.2}$$

Using (4.1) and (4.2), it is possible to obtain a Hamiltonian tour H such that

$$\Sigma_{e \in H} w(e) \geq max\{\frac{3}{4} W_2' + \frac{5}{6} W_3', W_2'\} \geq \frac{10}{13} \cdot OPT$$

Since $W_2' + W_3' \geq OPT$.

CHAPTER 5

SYMMETRIC MIN-TSP

5.1 Introduction

Let $G = (V, E, w)$ be an undirected graph, where w is a weight function associating a non-negative weight with each edge in E. Symmetric MIN-TSP is the classical problem of finding a minimum weight hamiltonian tour. As was mentioned earlier the problem is $NP - hard$ to approximate and thus there is no approximation algorithm for this problem unless $P = NP$. The problem does allow an approximation of 3/2 [2] when there is an added constraint that the edges in the graph obey triangle inequality. Another important special case is when the edge weights in the graph are drawn from the set $\{1, 2\}$. This special case of TSP was used by Karp [1] in his reduction to show that the problem is $NP - Complete$. It is easy to see that the edges automatically obey the triangle inequality and so a 3/2-approximation follows directly. A well known technique of subtour patching can be used to obtain an easy 5/4-approximation for the problem. First find a minimum weight cycle cover in the graph that has no 3-cycles (see Hartvigsen [7]). There are at most $n/4$ cycles in the graph. Pick any two cycles and combine them into one cycle with an increase of at most 2 in the total cost. Now combine this cycle with another cycle with an increase of at most 1 in the total cost. Continue in this fashion and obtain a tour with cost at most $5/4 \cdot OPT$. This patching technique was modified by Papadimitriou and Yannakakis [17] to obtain a 7/6-approximation for the problem. The algorithm is presented.

5.2 7/6-approximation for MIN-TSP-$\{1, 2\}$

Let $G = (V, E, w)$ be an undirected graph, where w is a weight function associating a non-negative weight from the set $\{1, 2\}$ with each edge in E; Assume for now that the

graph has a hamiltonian tour of weight $n = |V|$. Using the algorithm developed by Hartvigsen [7], find a minimum weight cycle cover that does not contain any 3-cycles. Let $C = \{c_1, c_2....c_m\}$ be the set of cycles in the cycle cover. Consider the two node sets C and V and create a bipartite graph B such that an edge $e = (u, v) \in B$ if there is an edge of weight 1 in the original graph from a node u in a cycle c and a node v that does not belong to c. Since there is a hamiltonian cycle with all edges of weight 1, it is easy to see that there is a matching in B. Obtain such a matching in B and then create a directed graph $F = (C, A)$ where $(c, c') \in A$ if c is matched with a node of c'.

Lemma 11 *F has a spanning subgraph S consisting of only in-trees of depth 1 and paths of length 2.*

Proof. Any weakly connected component of F has a cycle and each node of the cycle may have in-trees converging into it. Pick any node with an in-tree in the cycle and let l be the leaf node farthest from it. If $l's$ child $s$is not on the cycle then an in-tree is created by s and all it's parents and then s is removed. Continuing in this fashion, a cycle can be obtained with certain nodes having parent nodes outside the cycle. Pick a node with a parent outside the cycle and remove it with the parent. The parent of this node inside the cycle is included in it's in-tree if this leaves an even number or zero node between this node and any previous such node. This process leaves a cycle with no nodes outside the cycle and this can trivially be decomposed into paths of length one and at most one path of length 2. □

Now consider the spanning subgraph S and the in-trees of depth one. Any such in-tree has a root node which is a cycle, call it c and parent nodes that are also cycles, let these be $\{c_1, c_2, ...c_r\}$. It is easy to to see that all these cycles are connected to the root cycle c at different nodes in c. This is so because a directed edge from the matching has different end nodes. Now consider the cycle c and traverse it's nodes in
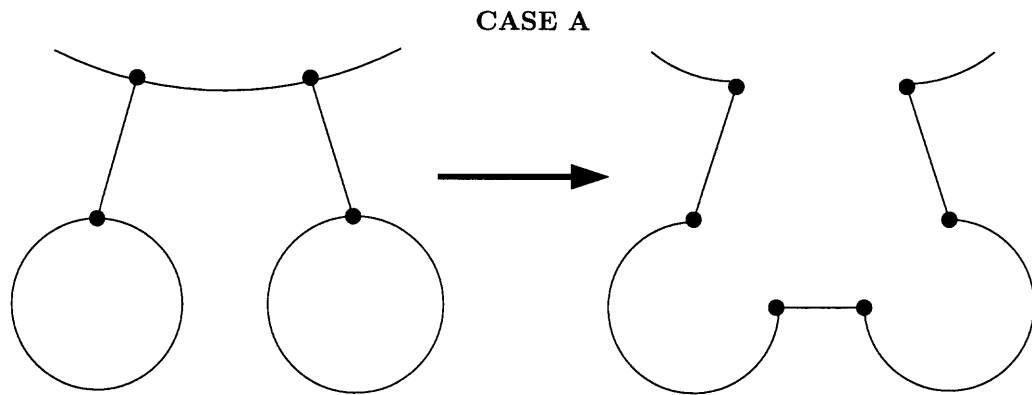
CASE A

Figure 5.1 (CASE A) Merging cycles.
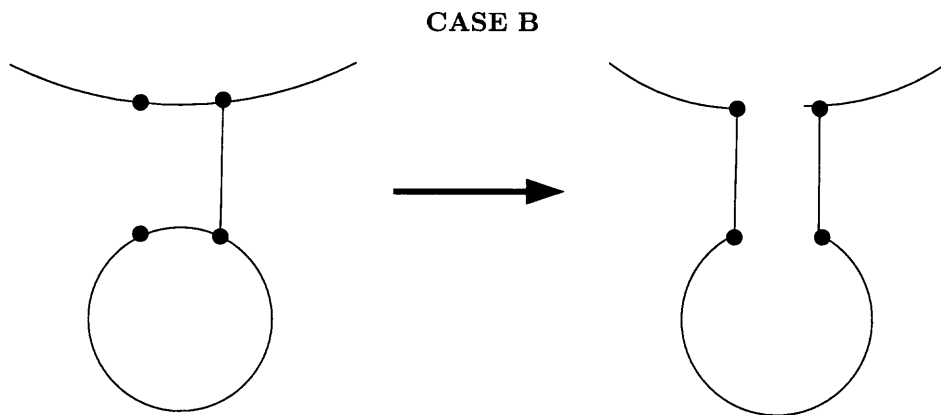
CASE B

Figure 5.2 (CASE B) Merging cycles.

clockwise fashion. If a node v is connected to a cycle, then check the next node. If this node is also connected to the cycle, then the three cycles are merged as shown in figure A, else the two cycles are merged as shown in figure B. In case of paths of length two, they are merged as shown in the figure C. The obtained cycles are then merged together using standard subtour patching technique. Let the tour thus obtained be T.

Lemma 12 T *has weight at most* $7/6 \cdot OPT$.

Proof. The total cost of the tour is the cost of the cycle cover plus the cost of merging the cycles. The cycle cover has cost n. Next it is shown that the cost of each
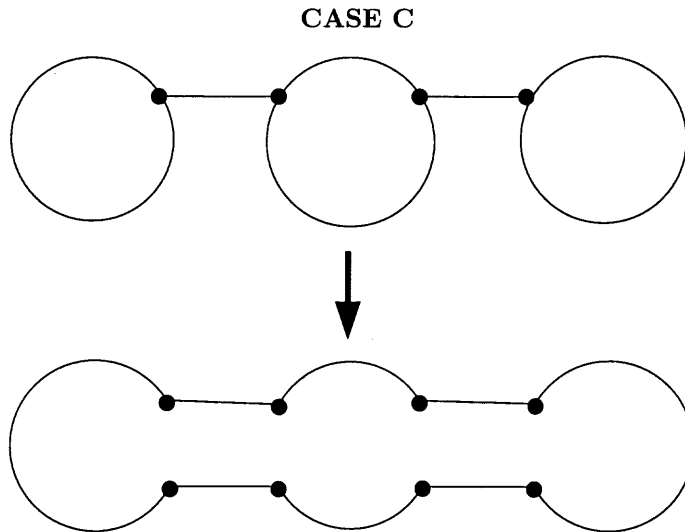
CASE C

Figure 5.3 (CASE C) Merging cycles.

of the merging is at most 1/6 per node. If the merging is done according to figure A, then the cost is at most 1/10 per node, since 10 nodes are involved in the process and the weight addition is at most 1. In case of figure B, it is easy to see that the cost is at most 1/6 per node and so is the case with figure C. $\qquad\square$

Next consider the case when the optimum tour has weight greater than n. Find a cycle cover in the graph without any 3-cycles as before. Merge all cycles that have an edge of weight 2 without any additional cost. Call this cycle *non-pure* and the remaining cycles *pure*. Let (u,v) be an edge of weight 2 in the non-pure cycle. If there is an edge of weight 1 connecting either u or v to a pure cycle then this cycle is merged with the non-pure cycle with no additional cost.

Now a bipartite graph B is constructed as before but the set C contains only the pure cycles. Again a directed graph F is obtained as before, but the matching in B may not be perfect this time. Some pure cycles may not have been matched. The non-pure cycle may or may not have some incoming edges into it.

Next F is decomposed into paths of length at most 2 and in-trees of depth 1. The decomposition may not be spanning this time. The components thus obtained

are merged as before with a cost of at most 1/6 per node in these components. Next the unmatched pure cycles are merged with a cost of 1 per cycle. Finally, the non-pure cycle is merged with no additional cost (if this was not already merged). Let the tour thus obtained be T'.

Lemma 13 T' *has weight at most* $7/6 \cdot OPT$, *and thus there is a* $7/6$-*approximation for MIN-TSP-$\{1, 2\}$.*

Proof. Let the number of edge of weight 2 in the cycle cover be k. Therefore the optimal TSP tour has weight at least $n + k$. Let the number of edges of weight 2 in the optimal tour be k'. Therefore the optimal tour has weight $n + k'$. Let r_2 be the unmatched pure cycles in F. It is easy to see that $r_2 \leq k'$. Let n_2 be the total number of nodes in these cycles. The cost of merging the decomposed components in F is at most 1/6 per node. The number of such nodes is at most $n - k - n_2$. This is so because the nonpure cycle has at least k edges of weight 2. And the end vertices of these edges are never involved in the merging process because there cannot be a weight one edge connecting such a vertex to a pure cycle.

Therefore the total cost of the tour is given by

$$cost \; \leq \; n \; + \; k \; + \; \frac{1}{6}(n \; - \; k \; - \; n_2)\, r_2$$

Now since $r_2 \leq n/4, k'$, therefore

$$cost \; \leq \; \frac{7}{6}n \; + \; \frac{5}{6}k \; + \; \frac{1}{3}k'$$
$$\leq \; \frac{7}{6} \cdot max\{(n+k), (n+k')\}$$
$$\leq \; \frac{7}{6} \cdot OPT$$

$\square$

CHAPTER 6

ASYMMETRIC MIN-TSP

6.1 Introduction

Given a directed graph $G = (V, E, w)$, where w is a weight function associating a non-negative weight with each edge in E; ATSP (Asymmetric MIN-TSP) is the problem of finding a minimum weight hamiltonian tour. This chapter focuses on metric-MIN-ATSP ie. the same problem with the added constraint that the edge weights observe the triangle inequality. Frieze, Galbiati and Maffioli [18] gave the first non-trivial approximation algorithm for the problem achieving a $\log(n)$- approximation. The result withstood any improvements for two decades until the recent paper by Blaeser [19] that achieves a tiny but insightful improvement, giving a $0.999 \cdot \log(n)$-approximation. Kaplan et al. [5] then improved this result achieving a $0.841 \cdot \log(n)$-approximation.

6.2 $0.841 \cdot \log(n)$-approximation for metric-MIN-ATSP

Presented here is the procedure developed by Kaplan et al. to achieve the above mentioned approximation. The procedure assumes that a subgraph G_c has been obtained from G using the procedure presented in Section 2.2. Also that G_c contains two cycle covers C_1 and C_2 with the following properties.

1. C_1 and C_2 do not share a 2-cycle, ie. if a 2-cycle is a part of C_1 then C_2 does not contain at least one of the edges from that 2-cycle.

2. The total weight of the two cycle covers is at most twice the weight of optimal TSP tour ie. $\Sigma_{e \in C_1} w(e) + \Sigma_{e \in C_2} w(e) \leq 2 \cdot OPT$

Let $N(G)$ denote the number of vertices, and $C(G)$ the number of connected components in a graph G. Let $G_1 = G_c$, $G_1' = C_1$ and $G_1'' = C_2$. From G_1, G_1'

and G_1'', choose the one that minimizes the value of $w(G)/\log(N(G)/C(G))$. Next contract all the connected components in this graph into vertices and delete all the self loops thus formed. Next find G_c in the new graph, set $G_2 = G_c$ and $G_2' = C_1$ and $G_2'' = C_2$ and apply the same procedure again. Proceed recursively till one single supervertex is obtained. Next uncontract the supervertices and add the edges from the corresponding connected components. Let the resulting graph be G_0. This graph contains a Euler cycle. Shortcut this cycle as in [2] to obtain the final tour T.

To analyze the performance of the algorithm, some properties are given in the following lemma.

Lemma 14 *At every recursive step i of the algorithm, $w(G_i) = w(G_i') + w(G_i'') \leq 2 \cdot OPT$ and $N(G_i) = N(G_i') = N(G_i'')$.*

Next, a bound is obtained on the total number of connected components in the graph at any recursive step in the algorithm.

Lemma 15 $C(G_i) + C(G_i') + C(G_i'') \leq N(G_i)$

Proof. A connected component has size k, if there are k vertices in this component. Let n_k denote the number of vertices and let I_k denote the number of components in connected components of size k in graph G_i. Obviously, $I_k = n_k/k$. Next, it is proved that total number of connected components in the three graphs G_i, G_i' and G_i'' is at most $\Sigma_k n_k$.

The analysis is divided into two cases. First consider all components of size k in G_i where k is odd. Since k is odd, therefore there is at least one cycle in G_i' and G_i'' that has length ≥ 3. Thus in the worst case both G_i and G_i'' have one cycle of length three and all other cycles are two cycles and so a total of at most $(k-1)/2$ cycles. Thus the total number of cycles in G_i' and G_i'' combined in components of size k is at most $\frac{n_k}{k} \cdot 2 \cdot \frac{k-1}{2}$. Now total number of connected components of size k

in G_i is n_k/k. Therefore total number of connected components of size k is at most $\frac{n_k}{k} \cdot (1 + 2 \cdot \frac{k-1}{2}) = n_k$.

Next consider connected components in G_i where k is even. Now it is not possible that all the cycles in the graphs G_i' and G_i'' are two cycles. Therefore, w.l.o.g., assume that G_i' has a cycle of length grater than 2. Now in the worst case G_i' will have either one cycle of length 4 or two cycles of length 3. In either case the total number of cycles in G_i' is at most $\frac{k-2}{2}$. The maximum number of cycles in G_i'' could however be $\frac{k}{2}$. It is easy to see as before that total number of connected components of size k is at most $\frac{n_k}{k} \cdot (1 + \frac{k-2}{2} + \frac{k}{2}) = n_k$. $\square$

Let the total number of steps in the algorithm be $P + 1$. Let $N_i = N(G_i)$ be the number of vertices and let w_i denote the total weight of the chosen graph at the ith iteration. The total cost of the final Euler tour is obviously $\Sigma i = 1^P w_i$. Which implies

$$\frac{\Sigma_{i=1}^P w_i}{\log(N_1)} = \frac{\Sigma_{i=1}^P w_i}{\Sigma_{i=1}^P \log(N_i/N_{i+1})} \leq max_{i=1,...,P} \frac{w_i}{\log(N_i/N_{i+1})} \qquad (6.1)$$

Let i be the iteration where the maximum of the right hand side is obtained. Here N_{i+1} is equal to the number of connected components in the graph that was chosen in the ith iteration. This implies

$$\begin{aligned}
\frac{w_i}{\log(N_i/N_{i+1})} &\leq min\{\frac{w(G_i)}{\log(N(G_i)/c(G_i))}, \frac{w(G_i')}{\log(N(G_i')/c(G_i'))}, \frac{w(G_i'')}{\log(N(G_i'')/c(G_i''))}\} \\
&\leq \frac{w(G_i) + w(G_i') + w(G_i'')}{\log(N^3(G_i)/c(G_i) \cdot c(G_i') \cdot c(G_i''))} \\
&\leq \frac{4 \cdot OPT}{3 \cdot \log_2 3} \\
&= 0.841 \cdot OPT \qquad\qquad (6.2)
\end{aligned}$$

The last inequality follows from the fact that $c(G_i) \cdot c(G_i') \cdot c(G_i'')$ is maximized when $c(G_i) = c(G_i') = c(G_i'') = N(G_i)/3$. From (6.1) and (6.2) it obtains,

$$\Sigma_{i=1}^{P} w_i \;\; \leq \;\; log_2 N_1 \cdot 0.841 \cdot OPT \;\; = 0.841 \cdot log_2 n \cdot OPT.$$

Since $N_1 = n$.

CHAPTER 7

RELATIONSHIP BETWEEN TSP AND CONNECTIVITY

7.1 Introduction

Given a graph $G = (V, E, w)$, such that w is a weight function that associates a non-negative weight with each edge in E. Let G' be a subgraph of G. G' is said to be *biconnected*, if the removal of any vertex and the corresponding incident edges from G', still leaves it connected. G' is said to be *2-edge-connected* if the removal of any edge from G' still leaves it connected. If all the edges in the graph have equal weights, then the problem of finding a minimum weight biconnected or 2-edge-connected subgraph has an efficient solution (Specifically, there are $\mathcal{O}(V+E)$ algorithms for both problems [20]). However, if the edges have unequal weights, then the problems are $NP-complete$ [20]. The relationship between TSP and minimum weight biconnected subgraph was first studied by Frederickson and Jaja [21]. They showed that when the edges in the graph obey the triangle inequality, then the weight of an optimal TSP tour is not greater than 3/2 times the weight of an optimal biconnected subgraph. Now suppose that the weight function w assigns edges from the set $\{1, 2\}$ to the edges E. The connectivity problems still remain $NP - complete$. Next it is shown that for this special class of graphs, the optimal TSP tour has weight exactly equal to the optimal 2-edge connected subgraph. And thus the result by Papadimitriou and Yannakakis [17] implies that there is a 7/6-approximation algorithm for finding a minimum weight 2-edge-connected subgraphs for this class of graphs.

7.2 2-edge-connected subgraph and TSP

Let $G = (V, E, w)$ be a complete undirected graph , such that w is a weight function that associates a weight from the set $\{1, 2\}$ with each edge in E. A graph is 2-edge-connected if there are at least two edge-disjoint paths between every pair of vertices in

the graph. 2-edge-connectivity is equivalent to the existence of an *ear decomposition* that is a partition of the edges of the graph into a sequence of ears (simple paths and cycles).

Let S be a minimum weight 2-edge-connected subgraph of G. Let T be the optimal TSP tour in G. Let $w(S)$ and $w(T)$ denote the total weight of the edges in the two subgraphs. The following lemma holds.

Lemma 16 *Let $G = (V, E, w)$ be an undirected complete graph with edge weight in $\{1, 2\}$. The weight of an optimal TSP tour T is exactly equal to the weight of a minimum weight 2-edge-connected subgraph S.*

Proof. Since T is a 2-edge-connected subgraph therefore it is obvious that $w(T) \geq w(S)$. Next, it is to be proved that $w(T) \leq w(S)$. Consider any 2-edge connected subgraph S' in G. Then it is sufficient to show that TSP tour can be obtained from S' without adding any additional weight.

Since S' is 2-edge-connected therefore S' has an ear decomposition. The first ear in such a decomposition is a single vertex. The start and end of each successive ear should be vertices occurring in previous ears, but all other vertices in an ear should be new. Such a decomposition can be found one ear at a time. Start each ear by any unused edge e from an already-explored vertex, and continue by a shortest path back to another already-explored vertex (such a path must exist because e cannot disconnect the graph). Therefore, the first ear in this decomposition is a vertex and the next ear is a cycle. Let this cycle be C. Now the next ear will have it's start and end vertices from this cycle. If it is possible to merge C and the next ear into a cycle C', then the next ear would have it's start and end vertices from C'. Again these two can be merged to obtain a cycle and then continuing in this fashion a tour can be obtained.
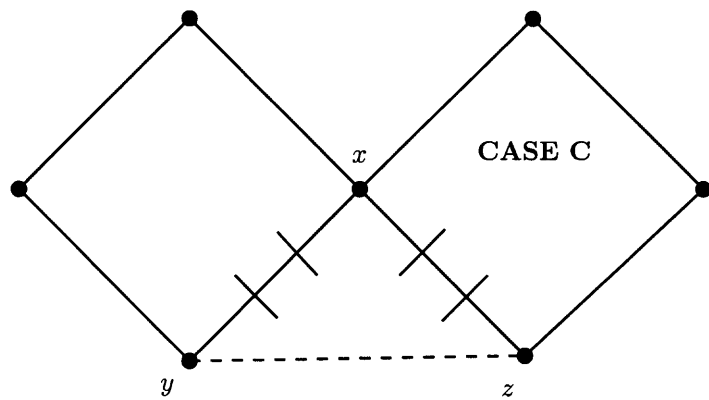
There are three possible ways in which a cycle and an ear can be connected. The first case (CASE A) is when the ear hits the cycle on consecutive vertices. Let
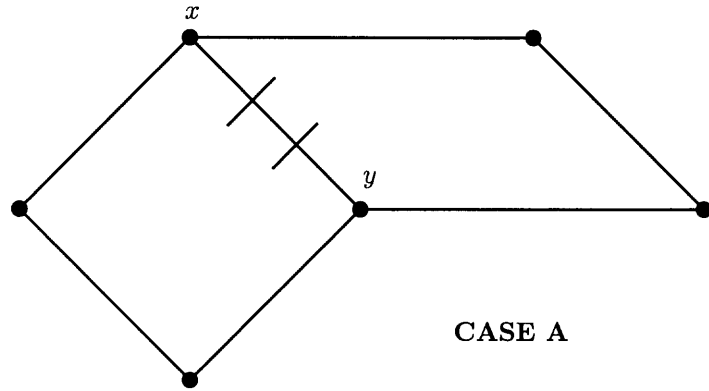
Figure 7.1 (CASE A) Merging two ears into a cycle.
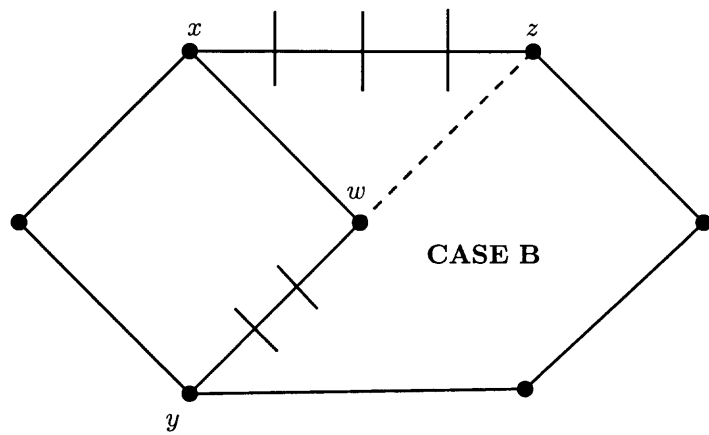
Figure 7.2 (CASE B) Merging two ears into a cycle.

Figure 7.3 (CASE C) Merging two ears into a cycle.

these vertices be x and y. Then the edge joining x and y is deleted and a single cycle is obtained. It is obvious that there is no loss of weight in this case. CASE B is when the ear hits the cycle on any non-consecutive vertices. Once again let these vertices be x and y. Give the cycle anti-clockwise orientation and clockwise orientation to the ear. Let vertex w be the successor to y on the cycle. Then the edge (y, w) is deleted. Let vertex z be the successor to x on the ear. Then the edge (z, x) is deleted. Next the edge (z, w) is added and a cycle is obtained. (Note that it is possible that there are no vertices on the ear, which means that z and y are the same vertex). Now to obtain a single cycle, two edges were deleted and a single edge was added. Since the edge weights are from the set $\{1, 2\}$, therefore there is no loss of weight. The third case, (CASE C) is when the two ear are connected at the same vertex. Let this vertex be x. Give a clockwise orientation to one ear and anti-clockwise orientation to the other ear. Let vertex y be the successor to x in the ear with the clockwise orientation. Then the edge (x, y) is deleted. Let vertex z be the successor to x in the ear with anti-clockwise orientation. Then the edge (z, x) is deleted. Next the edge (z, y) is added. By similar argument as in CASE B, there is no loss of weight in this case. $\square$

CHAPTER 8

CONCLUSION

Approximation algorithms first appeared as the computer science community's answer to the impossiblity of efficiently solving $NP-hard$ problems. Classical TSP however, belongs to the notorious class of problems that do not allow any approximation. Christofides [2] with his celebrated result in 1976 showed that when the edge weights in a graph obey triangle inequality, it is possible to obtain a 3/2-approximation. This was followed by a series of approximation results for a number of variants of TSP. The early results were mostly simple algorithms, whereas a lot of recent results have been obtained with algorithms that are more involved. Also linear programming has been used in addition to purely combinatorial techniques for some of the recent results. The fact that improvements had been achieved on this front very recently motivated this study. Alas, no results were obtained and it seems fair to conclude that nothing can be trivially improved. The more complicated techniques that were applied also did not yield any results or give any insights into how something could be improved. Recently Chen, Wang, Nagoya and Okamoto achieved some improvements for MAX-TSP and metric-MAX-TSP (The papers are as yet unpublished). The techniques applied to get these improvements are fairly involved but it is encouraging to see that there is still some hope for improvements on these results.

APPENDIX A

GRAPH THEORY GLOSSARY

This appendix contains graph theory terminology that is largely standard and could be useful in reading this thesis. Assume that $G = (V, E, w)$ is an undirected weighted graph, unless specified otherwise.

- **Bipartite Graph** - A graph G is bipartite if it's vertices can be partitioned into two disjoint subsets U and V such that each edge connects a vertex from U to one from V.

- **Cycle Cover** - A subgraph of G that is a collection of vertex disjoint cycles and covers all vertices of G.

- **Euler Circuit** - A close trail in a graph that includes every edge of the graph exactly once. The trail starts and end at the same vertex.

- **k-Cycle** - A cycle of length exactly k.

- **k-Cycle Cover** - A cycle cover of G such that every cycle has length at least k.

- **k-Path Coloring** - A k-coloring of G such that edges from any given color form a collection of vertex disjoint paths.

- **Triangle Inequality** - If the edges in G observe the triangle inequality, then $w(u, v) \leq w(u, w) + w(v, w) \; \forall \; u, v, w \in E$

REFERENCES

[1] R. M. Karp, *Reducibility among combinatorial problems, In R. E. Miller and J. W. Thatcher, editors, Complexity of Computer Computations.* New York: Plenum Press, 1972.

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

[3] R. Hassin and S. Rubinstein, "Better approximation algorithms for Max TSP," *Information Processing Letters,* 2000.

[4] R. Hassin and S. Rubinstein, "A 7/8 approximation algorithm for metric Max TSP," *Information Processing Letters,* 2002.

[5] H. Kaplan, M. Lewenstein, N. Shafrir, and M. Sviridenko, "Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs," *FOCS,* 2003.

[6] M. Blaeser, "A 3/4 approximation algorithm for maximum asymmetric TSP with weights zero and one," *APPROX,* 2004.

[7] D. Hartvigsen, "Extensions of matching theory," *Ph.D. Dissertation, Carnegie Mellon University,* 1984.

[8] M. Lewenstien and M. Sviridenko, "Approximating asymmetric maximum TSP (preliminary version)," *SODA,* 2003.

[9] N. Alon, "A simple algorithm for edge-coloring bipartite multigraphs," *Information processing letters,* pp. 301–302, 1985.

[10] A. I. Serdyukov, "An algorithm with an estimate for the traveling salesman problem of the maximum," *Upravlyaemye Sistemy (In Russian),* pp. 85–103, 1984.

[11] A. V. Kostochka and A. I. Serdyukov, "Polynomial algorithms with the estimates $\frac{3}{4}$ and $\frac{5}{6}$ for the traveling salesman problem of the maximum(Russian)," *Upravlyaemye Sistemy 26,* pp. 55–59, 1985.

[12] R. Hassin and S. Rubinstein, "An approximation algorithm for maximum traveling salesman problem," *Information Processing Letters, 67,* pp. 125–130, 1998.

[13] M. L. Fisher, L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for finding a maximum weight Hamiltonian circuit," *Networks,* pp. 799–809, 1979.

[14] S. R. Kosaraju, J. K. Park, and C. Stein, "Long tours and short superstrings," *FOCS,* 1994.

[15] M. Blaeser, "An $\frac{8}{13}$ approximation algorithm for the asymmetric Max-TSP," *SODA*, pp. 64–73, 2002.

[16] D. Breslauer, T. Jiang, and Z. Jiang, "Rotations of periodic strings and short superstrings," *Journal of Algorithms*, pp. 24(2):340–353, 1997.

[17] C. H. Papadimitriou and M. Yannakakis, "The traveling salesman problem with distances one and two," *Mathematics of Operations Research 18*, pp. 1–11, 1993.

[18] A. M. Frieze, G. Galbiati, and F. Maffioli, "On the worst-case performance of some algorithms for the asymmetric traveling salesman problem," *Networks, 12(1)*, pp. 23–39, 1982.

[19] M. Blaeser, "A new approximation algorithm for the asymmetric TSP with triangle inequality," *SODA*, 2003.

[20] K. P. Eswaran and R. E. Tarjan, "Augmentation problems," *SIAM Journal of Computing*, pp. 653–665, 1976.

[21] G. N. Frederickson and J. JaJa, "On the relationship between the biconnectivity augmentation and traveling salesman problem," *Theoretical Computer Science, 19(1982)*, pp. 189–201, 1982.