An $O(\log n)$ Approximation Ratio for the Asymmetric Traveling Salesman *Path* Problem

Chandra Chekuri* Martin Pál†

February 16, 2006

Abstract

Given an arc-weighted directed graph $G = (V, A, \ell)$ and a pair of vertices s, t, we seek to find an s-t *walk* of minimum length that visits all the vertices in V. If ℓ satisfies the *asymmetric* triangle inequality, the problem is equivalent to that of finding an s-t *path* of minimum length that visits all the vertices. We refer to this problem as ATSPP. When $s = t$ this is the well known asymmetric traveling salesman tour problem (ATSP). Although an $O(\log n)$ approximation ratio has long been known for ATSP, the best known ratio for ATSPP is $O(\sqrt{n})$. In this paper we present a polynomial time algorithm for ATSPP that has approximation ratio of $O(\log n)$. The algorithm generalizes to the problem of finding a minimum length path or cycle that is required to visit a subset of vertices $v_1, v_2, \ldots, v_k$ in a given order.

1 Introduction

In the classical traveling salesman problem (TSP) we are given an undirected (directed) graph with edge (arc) lengths and we seek to find a Hamiltonian cycle of minimum length. It is one of the most extensively studied combinatorial optimization problems. TSP is not only NP-hard, it is also NP-hard to approximate to within any polynomial factor - both these facts follow easily from the NP-Completeness of the Hamiltonian cycle problem. If one relaxes the constraint and asks for a tour instead of a cycle, that is a vertex can be visited multiple times, we obtain more tractable variants of the problem. In the undirected graph setting this is equivalent to assuming that the edge lengths satisfy the triangle inequality and in directed graphs this is equivalent to the arc lengths satisfying the asymmetric triangle inequality. These problems are referred to as Metric-TSP and ATSP respectively. For Metric-TSP the best known approximation is a factor of $3/2$ due to Christofides [6] and for ATSP the best known ratio is $O(\log n)$ first established by Frieze, Galbiati and Maffioli [8]. The smallest known approximation ratio is $0.842 \log_2 n$ [12] improving upon the $\log_2 n$ in [8] and $0.999 \log_2 n$ in [3].

In this paper we are concerned with the traveling salesman *path* problem. The input to the problem is a graph with edge/arc lengths and two vertices s and t. We seek a path from s to t of minimum length that visits all the vertices. As with TSP the path version is NP-hard and also hard to approximate via a reduction

*Lucent Bell Labs, 600 Mountain Avenue, Murray Hill, NJ 07974. chekuri@reserch.bell-labs.com
†Google Inc., 1440 Broadway, New York, NY 10018. mpal@google.com

from the Hamiltonian path problem. We therefore consider the relaxed version where we ask for a walk instead of a path. We refer to undirected and directed versions as Metric-TSPP and ATSPP respectively. For Metric-TSPP the best known approximation ratio is $5/3$ due to Hoogeveen [11] (see [10] for a different proof). The ATSPP problem does not seem to have been considered much in the literature and we are only aware of the recent work of Lam and Newman [14] who give an $O(\sqrt{n})$ approximation. Our main result is the following.

Theorem 1.1 *There is an $O(\log n)$ approximation algorithm for the* ATSPP *problem.*

We also consider a generalization of ATSPP. We are given a sequence of distinct nodes $v_1, v_2, \ldots, v_k$ and seek a minimum length path P (or cycle) that visits all nodes of the graph but visits $v_1, v_2, \ldots, v_k$ in that order. We can assume without loss of generality that the path P starts at v_1 and ends at v_k. In the undirected setting, this problem has been referred to as path-constrained TSP and is a special case of a more general problem called precedence-constrained TSP [4]. Recently Bachrach *et al.* [2] gave a 3-approximation for the path-constrained TSP in metric spaces. Here we show that our approach for ATSPP generalizes to the asymmetric version of the path-constrained TSP.

Theorem 1.2 *There is an $O(\log n)$ approximation algorithm for the path-constrained* ATSPP *problem.*

ATSPP vs ATSP: An α approximation algorithm for ATSPP yields the same approximation ratio for ATSP by picking an arbitrary vertex to be both the start and end vertex of the path. On the other hand an approximation algorithm for ATSP does not imply an algorithm for ATSPP. This is in contrast to the metric case; Metric-TSP and Metric-TSPP are approximable to within constant factors of each other. At first glance it appears that ATSPP can be reduced to ATSP by adding an arc (t, s) of appropriate length. However it is not to hard to convince oneself that such a reduction does not work. To better understand the difficulty in the directed setting and develop the main ingredient of our algorithm we give a brief overview of the algorithm of Frieze *et al.* [8] for ATSP and a variant proposed by Kleinberg and Williamson [13] (see [16] for a description and proof). Both algorithms work in an iterative fashion.

The algorithm in [8] finds a collection of directed cycles that partition the vertex set (called a cycle-cover in some settings) such that the total cost of the cycles is at most OPT. This can be done in polynomial time using a reduction to assignment problem. A vertex is chosen from each cycle to be its proxy and the problem is reduced to the graph induced on the proxy vertices. Note that the number of proxies is no more than half the number of initial vertices. A tour in the smaller graph can be extended to the full graph using the initial cycles. Further, it can be easily seen that there must be a tour of length OPT in the new instance as well. Thus the algorithm pays a cost of OPT in each iteration and since the number of vertices is reduced by a factor of 2 in each iteration the overall cost paid is bounded by $\log_2 n \cdot$ OPT.

The algorithm in [13] works differently. It finds a single cycle in each iteration where the ratio of the cost of the cycle to the number of vertices in the cycle is minimum. Such a cycle can be found in polynomial time. Again a proxy vertex is chosen from the cycle and the algorithm works in a reduced graph with the other vertices of the cycle removed. The analysis is similar to that for the greedy algorithm for covering problems, in particular the set cover problem [7]. This results in an approximation ratio of $2H_n$ where H_n is the n-th harmonic number.

Both the algorithms described rely on the fact that given a collection of cycles we can reduce the problem by choosing a representative from each cycle and ignoring the other vertices in the cycles. While cycles are useful in the ATSPP setting as well, they can no longer be relied on as sole building blocks. In addition to cycles, we also need to maintain a partial path from s to t. This restricts our choice of improvement steps: the only way to augment a partial path P that suggests itself is to replace one of the arcs (u, v) on P by a subpath P' from u to v that visits some yet unvisited vertices. Our main technical contribution is to show that given any partial path, there *exists* an augmentation to a feasible path such that the cost of augmenting is at most 2OPT. Here OPT denotes the length of an optimum solution. We combine this with the greedy approach similar to that in [13] to obtain desired algorithm.

Related Work: TSP is a cornerstone problem for combinatorial optimization and there is a vast amount of literature on many aspects including a large number of variants. The books [15, 9] provide extensive pointers as well as details. Our work is related to understanding the approximability of TSP and its variants. In this context one of the major open problems is to resolve whether ATSP has a constant factor approximation. The natural LP relaxation for ATSP has only a lower bound of 2 on its integrality gap [5]. Resolving the integrality gap of this formulation is also an important open problem. The path-constrained TSP problem is a special case of the precedence-constrained TSP problem [4]: we are given a partial order on the vertices and the goal is to seek a minimum length cycle that visits vertices in an order that is consistent with the given partial order. In [4] it is shown that this general problem is hard to approximate for even special classes of metric spaces.

2 Preliminaries

Let G be an arc-weighted directed graph. For a path P in G let $V(P)$ and $A(P)$ denote the vertices and arcs of P respectively. Let $\mathcal{P}(s,t)$ denote the set of all $s \rightsquigarrow t$ paths in G. A path $P \in \mathcal{P}(s,t)$ is *non-trivial* if it contains internal vertices, that is $|V(P)| > 2$. Let $\mathcal{C}(s,t)$ denote the set of cycles in G that do *not* contain either s or t. Let P be a non-trivial path in $\mathcal{P}(s,t)$. Then the *density* of P denoted by $d(P)$ is the ratio of the total arc length of P to the number of internal vertices in P. In other words $d(P) = \sum_{a \in A(P)} \ell(a)/|V(P) - 2|$. Similarly the density of a cycle $C \in \mathcal{C}(s,t)$ is defined to be $d(C) = \sum_{a \in A(C)} \ell(a)/|V(C)|$.

Lemma 2.1 *Given a directed graph G and two vertices s, t, let λ^* be the density of the minimum density non-trivial path in $\mathcal{P}(s,t)$. There is a polynomial time algorithm that either finds a path $P \in \mathcal{P}(s,t)$ such that $d(P) = \lambda^*$ or finds a cycle $C \in \mathcal{C}(s,t)$ such that $d(C) < \lambda^*$.*

Proof: Given a parameter $\lambda > 0$ we give a polynomial time algorithm that either finds a non-trivial path P with $d(P) \leq \lambda$ or a cycle $C \in \mathcal{C}(s,t)$ with $d(P) < \lambda$ or reports that no path or cycle has a density at most λ. This can be combined with binary search to obtain the desired algorithm.

We can assume without loss of generality that in G there are no arcs into s and no arcs out of t. This ensures that there are no cycles that contain s or t. Given λ we create a graph G_λ that differs from G only

in the arc weights. The arc weights of G_λ are set as follows:

$$\begin{aligned} \ell'(s,u) &= \ell(s,u) - \lambda/2 \qquad u \in V - \{s,t\} \\ \ell'(u,t) &= \ell(u,t) - \lambda/2 \qquad u \in V - \{s,t\} \\ \ell'(u,v) &= \ell(u,v) - \lambda \qquad u,v \in V - \{s,t\} \end{aligned}$$

It is easy to verify that the density of a path $P \in \mathcal{P}(s,t)$ or a cycle $C \in \mathcal{C}(s,t)$ is at most λ iff its length in G_λ is non-positive. Thus we can use Bellman-Ford algorithm to compute a shortest path in G_λ between s and t. If we find a negative cycle we are done. Otherwise, if the shortest path length is non-positive then we obtain a path of density at most λ. If the shortest path is positive we report the non-existence of a path or cycle of density λ. ■

We remark that the above proof only guarantees a weakly-polynomial time algorithm due the binary search for λ^*. However we remark that one could use a *parametric* shortest path algorithm to obtain a strongly-polynomial time algorithm. Our focus is on the approximation ratio and hence we do not go into the details of this well-understood area and refer the reader to [1, 17].

Given a directed path P and two vertices $u, v \in P$ we write $u \preceq_P v$ if u precedes v in P (we assume that u precedes itself). If $u \preceq_P v$ and $u \neq v$ we write $u \prec_P v$. If P is clear from the context we simply write $u \preceq v$ or $u \prec v$.

We call a path $P \in \mathcal{P}(s,t)$ *spanning* if $V(P) = V$, otherwise it is *partial*. Let P_1 and P_2 be two paths in $\mathcal{P}(s,t)$. We say that P_2 *dominates* P_1 iff $V(P_1) \subset V(P_2)$. We say that P_2 is an *extension* of P_1 if P_2 dominates P_1 and the vertices in $V(P_1)$ are visited in the same sequence in P_2 as they are in P_1. It is clear that if P_2 extends P_1 then we can obtain P_2 by replacing some arcs of P_1 by subpaths of P_2. Let $\ell(P_1, P_2)$ denote the *cost of extension* which is defined to be $\sum_{a \in A(P_2) \setminus A(P_1)} \ell(a)$. Note that the cost of extension does not include the length of arcs in P_1.

3 Augmentation Lemma

Our main lemma is the following.

Lemma 3.1 *Let $G = (V, A, \ell)$ satisfy the asymmetric triangle inequality and let P_1, P_2 in $\mathcal{P}(s,t)$ such that P_2 dominates P_1. Then there is a path $P_3 \in \mathcal{P}(s,t)$ that dominates P_2, extends P_1, and satisfies $\ell(P_1, P_3) \leq 2\ell(P_2)$.*

We remark that the above lemma only guarantees the existence of P_3 but not a polynomial time algorithm to find it. Let us introduce some syntactic sugar before plunging into the proof. For a path P and two vertices $u \preceq_P v$ on P, we use $P(u,v)$ to denote the subpath of P starting at u and ending at v. Specifically for the path P_1, we use the following notation: for a vertex $u \in P_1 \setminus \{t\}$, we denote by u^+ the successor of u on P_1.

Proof of Lemma 3.1: Consider the set $X \subseteq P_1$ of vertices u with the property that $u \prec_{P_2} u^+$. For each such vertex, we think of replacing the arc (u, u^+) of P_1 by the subpath $P_2(u, u^+)$. Naïvely, we could replace all arcs (u, u^+) by the corresponding subpaths of P_2. Unfortunately this might cause some arcs of P_2 to be used multiple times and thus incur high cost. To avoid paying this cost, we choose only some of the vertices

in X to replace their corresponding arcs. We shall *mark* a subset of vertices $u \in X$ with their corresponding path segments $P_2(u, u^+)$ such that each vertex of P_2 occurs in some marked path segment at least once, while each arc of P_2 appears in at most *two* marked segments.

We construct a sequence $g_1, g_2, \ldots$ of marked vertices iteratively. To start, we let $g_1 = s$ be the first marked vertex. Given $g_1, \ldots, g_i$, we construct g_{i+1} as follows. Find the last vertex v on the subpath $P_1(g_i^+, t)$ such that $v \in P_2(s, g_i^+)$. Such a vertex v always exists, as g_i^+ belongs to both path segments. Note that $v^+ \notin P_2(s, g_i^+)$, which means that (unless $v = t$) $v \prec_{P_2} v^+$ and thus $v \in X$. If $v \neq t$, we let $g_{i+1} = v$ and continue to the next iteration. If $v = t$, we stop. Let g_l be the last vertex of the constructed sequence. To prove the lemma, it now suffices to prove the following two statements.

(P1) For every vertex $v \in P_2$, there is at least one marked segment $P_2(g_i, g_i^+)$ that contains v.

(P2) Every arc $a \in P_2$ belongs to at most two marked segments $P_2(g_i, g_i^+)$, with $i = 1, \ldots, l$.

These statements in turn follow from the following inequalities:

(I1) For $i = 1, \ldots, l-1$, we have $g_i \prec_{P_1} g_{i+1}$.

(I2) For $i = 1, \ldots, l-1$, we have $g_i \prec_{P_2} g_{i+1} \preceq_{P_2} g_i^+$.

(I3) For $i = 1, \ldots, l-2$, we have $g_i^+ \preceq_{P_2} g_{i+2}$.

In particular, (I2) shows that any two consecutive path segments $P_2(g_i, g_i^+)$ and $P_2(g_{i+1}, g_{i+1}^+)$ overlap. Since the first segment contains s and the last segment contains t, the union of these segments must necessarily cover the whole path P_2. Hence (P1) holds. Inequalities (I2) and (I3) imply that two path segments $P_2(g_i, g_i^+)$ and $P_2(g_j, g_j^+)$ overlap only if $|i - j| \leq 1$, and thus each arc $a \in P_2$ can belong to at most two consecutive segments. This proves (P2).

We finish the proof by showing that (I1)–(I3) hold. (I1) holds by construction, as $g_{i+1} \in P_1(g_i^+, t)$. The second part of (I2), $g_{i+1} \preceq_{P_2} g_i^+$ is easily seen to hold as well, since g_{i+1} is defined to be the last vertex v along the path P_1 such that $v \preceq_{P_2} g_i^+$.

From (I1) we know that g_{i+2} occurs on the path P_1 later than g_{i+1}, thus it must be that $g_{i+2} \preceq_{P_2} g_i^+$ does not hold, and hence $g_i^+ \prec_{P_2} g_{i+2}$. This proves inequality (I3).

Finally, we prove the first part of inequality (I2), $g_i \prec_{P_2} g_{i+1}$. Since $g_1 = s$, this certainly holds for $i = 1$. For contradiction, suppose that $g_{i+1} \preceq_{P_2} g_i$ for some $i > 1$. Consider the iteration in which g_i got marked. Recall that by construction, g_i is the last vertex along the path P_1 that belongs to $P_2(s, g_{i-1}^+)$. But then, from $g_{i+1} \preceq_{P_2} g_i$ and $g_i \preceq_{P_2} g_{i-1}^+$ it follows that $g_{i+1} \preceq_{P_2} g_{i-1}^+$, and hence $g_{i+1} \in P_2(s, g_{i-1})$. This is a contradiction, because by (I1), g_{i+1} occurs on P_1 later than g_i.

■

We obtain the following useful corollary.

Corollary 3.2 *Let $P \in \mathcal{P}(s,t)$. Then there is a spanning path $P' \in \mathcal{P}(s,t)$ such P' extends P and $\ell(P, P') \leq 2\textsc{Opt}$.*

Proof: In Lemma 3.1, we let $P_1 = P$ and we choose P_2 to be some fixed optimum spanning path. The path P_3 guaranteed by the lemma is the desired P'. ■

4 Algorithm for ATSPP

Our algorithm for ATSPP works in a greedy fashion, choosing a best ratio augmentation in every step similar in spirit to that in [13]. The approximation ratio follows from the same arguments as in the analysis of the greedy algorithm for set cover [7].

At any point in time, the algorithm maintains an s-t path $P = (s = p_0, p_1, \ldots, p_k = t)$ and a list $\mathcal{C}$ of disjoint cycles $C_1, \ldots, C_l$. The cycles are at all times disjoint from P and together with P partition the vertex set V. From each cycle C_i, we pick a vertex c_i as a proxy for that cycle. Initially, the path P consists of a single arc s-t, and every vertex $v \in V \setminus \{s, t\}$ is considered a separate (degenerate) cycle. (Thus initially, each vertex will be its own cycle's proxy.)

In each iteration, we seek to decrease the number of components by performing a *path/cycle augmentation*. In a path augmentation step, we pick a path π that starts at some vertex $p_i \neq t$ on the path P, visits one or more cycle proxy vertices, and ends at p_{i+1}, the successor of p_i on P. Let $R(\pi) = c_{i_1}, c_{i_2}, \ldots, c_{i_m}$ be the set of proxy vertices visited by π. Consider the union of the path π and the cycles $\{C_i\}_{c_i \in R(\pi)}$. In this graph, the in-degree of every vertex equals its out-degree, except for p_i and p_{i+1}. Thus, it is possible to construct an Eulerian walk from p_i to p_{i+1} that visits all arcs (and hence all vertices) of $\bigcup_{c_i \in R(\pi)} C_i$. Using triangle inequality and short-cutting, we convert the walk into a path π' that visits every vertex only once without increasing its cost. We then extend P by replacing the arc $p_i p_{i+1}$ by the path π'. Finally, we remove all cycles in $R(\pi)$ from $\mathcal{C}$.

The cycle augmentation step is very similar. We pick a non-degenerate cycle C on proxy vertices (that is, it contains two or more proxy vertices). We let $R(C)$ be the set of proxy vertices visited by C, and consider the graph $C \cup \bigcup_{c_i \in R(C)} C_i$. This graph is Eulerian: by following an Eulerian tour of it and short-cutting, we obtain a cycle C' visiting every vertex of $\bigcup_{c_i \in R(C)} C_i$. We don't know how to use C' to extend the path P; instead, we add C' to the list $\mathcal{C}$ (we pick a proxy for C' arbitrarily). Again, we remove all cycles in $R(C)$ from the list $\mathcal{C}$.

In every iteration, we pick a path or a cycle augmentation step with minimum density. In the following, we use π to refer to either an augmenting path or augmenting cycle. For the purposes of this algorithm, we define the density of a path or cycle π to be $d(\pi) = \ell(\pi)/|R(\pi)|$ the ratio of the length of π to the number of proxy vertices covered by π. Note that although we consider only proxy vertices in the above definition of density, we can still use Lemma 2.1 to find, in polynomial time, an augmenting path of minimum density λ^*, or find an augmenting cycle with density no greater than λ^*.

Each augmenting path or cycle iteration reduces the size of the list $\mathcal{C}$, and hence it takes at most $|V| - 2$ iterations to exhaust it. At this point, all outstanding cycles must have been included in P, and hence P must be a valid ATSPP path. We thus output P and stop.

4.1 Bounding the cost

We now turn to bounding the cost of the resulting path. To do this, we observe the quantity $L = \ell(P) + \sum_{c \in \mathcal{C}} \ell(C)$. Initially, $L = \ell(s, t) \leq \textsc{Opt}$. Note that in every augmentation step, L increases by at most $\ell(\pi)$, where π is the current augmenting path or cycle. Hence, it is enough to bound the cost of the augmenting paths/cycles.

Claim 4.1 *In every iteration, if π is the augmenting path or cycle in that iteration,*

$$\ell(\pi) \leq \frac{|R(\pi)|}{|\mathcal{C}|} \cdot 2\textsc{Opt}.$$

Proof: Let P^* be a minimum-cost s-t path that visits all proxy vertices of cycles in $\mathcal{C}$. One such path can be obtained by short-cutting the optimum ATSPP path, hence $\ell(P^*) \leq \textsc{Opt}$. Lemma 3.1 states that the path P can be extended to a path P_3 such that $R(C) \subseteq P_3$ and the cost of the extension is at most $2\ell(P^*) \leq 2\textsc{Opt}$. The extension covers $|\mathcal{C}|$ proxy vertices, and hence has density at most $2\textsc{Opt}/|\mathcal{C}|$. The subpaths of this extension are also valid augmentation paths; and one of them must have density no greater than the density of the whole extension. Thus, there is an augmenting path with ratio $2\textsc{Opt}/|\mathcal{C}|$; the density of the best path or cycle can only be lower. ■

Lemma 4.1 *The overall cost of the path output by the algorithm is at most* $\max(4H_{n-2}, 1) \cdot \textsc{Opt}$.

Proof: At any given stage of the algorithm, let $k = |\mathcal{C}|$ be the number of components left. We claim that (*) the cost of reducing k by one is at most $4\textsc{Opt}/k$. Summing over $k = 1, \ldots, |V| - 2$ yields a bound of $4H_{n-2}\textsc{Opt}$. We also have to account for the arc (s, t) purchased in the initialization phase; but note that if $n \geq 3$, this arc will be removed during the execution of the algorithm and hence does not contribute to the final cost. It is easy to verify that for $n = 2$, our algorithm finds an optimal solution.

To prove the claim (*), consider any fixed value of k and consider the augmentation step in which the value of $|\mathcal{C}|$ drops from some $k_1 \geq k$ to $k_2 < k$. The augmentation step was either a path step, or a cycle step. In a path step, $k_1 - k_2$ cycles are removed at cost $2\textsc{Opt}(k_1 - k_2)/k_1$, i.e. $2\textsc{Opt}/k_1 \leq 2\textsc{Opt}/k$ per cycle. In a cycle step, $k_1 - k_2 + 1$ cycles are removed and one cycle is added, at cost $2\textsc{Opt}(k_1 - k_2 + 1)/k_1$. The amortized cost per cycle is thus $2\frac{\textsc{Opt}}{k_1} \cdot \frac{k1-k_2+1}{k_1-k_2}$. Since in a cycle step, $k_1 - k_2 \geq 1$, the amortized cost per cycle is at most $4\textsc{Opt}/k_1$. ■

We briefly discuss the running time of the algorithm. The number of augmenting iterations is, in the worst case, linear in n. In each iteration we need to find a parametric shortest path between every adjacent pair of vertices in the current partial path. Thus, in the worst case the algorithm requires $\Theta(n^2)$ parametric shortest path computations. Each parametric shortest path computation can be implemented in $O(nm + n^2 \log n)$ time in a graph with n nodes and m arcs [17]. One way to simplify the implementation is to use the transitive closure of the original graph: an arc (u, v) in the trantive closure has length equal to the shortest path from u to v in the original graph. A simple upper bound on the number of arcs in the closure is n^2. Thus a parametric shortest path computation takes $O(n^3)$ time. Putting together these bounds, we can implement our algorithm in $O(n^5)$ time. The running time can be improved at the expense of a (slightly) worse approximation guarantee. In particular the density computation for the augmentation in each iteration can be approximate.

Path-constrained ATSPP: We show that our algorithm for ATSPP generalizes to the path-constrained version. Recall that we are given a sequence of vertices $s = v_1, v_2, \ldots, v_k = t$ and seek a minimum length spanning path in $\mathcal{P}(s, t)$ that visits $v_1, v_2, \ldots, v_k$ in order. The only change from the algorithm for ATSPP is in the initialization step. Instead of starting with a path consisting of the arc (s, t) we start with a path P consisting of the arcs $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$. Note that the cost of this path is a lower bound on

the cost of an optimum path. We can apply the augmentation lemma from now on and proceed as before. The analysis is essentially the same as for ATSPP.

Acknowledgments: We thank Fumei Lam for an enlightening conversation, for sending us a copy of the manuscript [14] and for pointing out [2]. We thank Moses Charikar for pointing out [13]. Part of this work was done while the second author was at Lucent Bell Labs. Chandra Chekuri acknowledges support from an ONR basic research grant N00014-05-1-0256 to Lucent Bell Labs.

References

[1] R. Ahuja, T. Magnanti and J. Orlin. Network Flows. Prentice Hall, 1993.

[2] A. Bachrach, K. Chen, C. Harrelson, S. Rao and A. Shah. Lower Bounds for Maximum Parsimony with Gene Order Data. *RECOMB Comparative Genomics*, 1–10, 2005.

[3] M. Bläser. A New Approximation Algorithm for the Asymmetric TSP with Triangle Inequality. *Proc. of ACM-SIAM SODA*, 638–645, 2002.

[4] M. Charikar, R. Motwani, P. Raghavan and C. Silverstein. Constrained TSP and lower power computing. *Proc. of WADS*, 104–115, 1997.

[5] M. Charikar, M. Goemans, and H. Karloff. On the Integrality Ratio for Asymmetric TSP. *Proc. of IEEE FOCS*, 101–107, 2004.

[6] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Technical report, GSIA, CMU, 1976.

[7] V. Chvatal. A greedy heuristic for the set-covering problem. *Math. of Oper. Res.*, Vol 4:233–235, 1979.

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

[9] G. Gutin and A. P. Punnen (Eds.). Traveling Salesman Problem and Its Variations. Springer, Berlin, 2002.

[10] N. Guttmann-Beck, R. Hassin, S. Khuller and B. Raghavachari. Approximation Algorithms with Bounded Performance Guarantees for the Clustered Traveling Salesman Problem. *Algorithmica*, Vol 28 pp. 422–437, 2000. Preliminary version in *Proc. of FSTTCS*, 1998.

[11] J. Hoogeveen. Analysis of Christofides' heuristic: Some paths are more difficult than cycles. *Operations Research Letters*, 10:291–295, 1991.

[12] H. Kaplan, M. Lewenstein, N. Shafir and M. Sviridenko. Approximation Algorithms for Asymmetric TSP by Decomposing Directed Regular Multidigraphs. *Proc. of IEEE FOCS*, 56–67, 2003.

[13] J. Kleinberg and D. Williamson. Unpublished note, 1998.

[14] F. Lam and A. Newman. Traveling Salesman Path Problems. Manuscript, April 2005.

[15] E. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. Shmoys (Eds.). The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. John Wiley & Sons Ltd., 1985.

[16] D. Williamson. Lecture Notes on Approximation Algorithms. IBM Research Report RC 21273, February 1999.

[17] N. Young, R. Tarjan and J. Orlin. Faster parametric shortest path and minimum balance algorithms. *Networks*, 21(2): 205–221, 1991.