

Approximation Algorithms for Deadline-TSP

Nikhil Bansal* Avrim Blum* Shuchi Chawla* Adam Meyerson†

November 4, 2003

Abstract

Give a metric space G on n nodes, with a start node r , and a deadline $D(v)$ for each vertex v , the Deadline-TSP problem is to find a path starting at r that visits as many nodes as possible within their deadlines. This is a classic optimization problem with no known good approximations for general metric spaces. We give an $O(\log n)$ approximation algorithm for this problem, and use as a subroutine a constant-factor approximation that we develop for a generalization of the orienteering problem in which both the start and the end nodes of the path are fixed. We also give a bicriteria approximation algorithm for Deadline TSP: Given an $\epsilon > 0$, our algorithm produces a $\log(1/\epsilon)$ approximation, while exceeding the deadlines by a factor of $1 + \epsilon$. In the process, we also give a 3-approximation to the orienteering problem, improving on the previously best known 4-approximation of [BCK⁺03].

1 Introduction

Consider a delivery-person that starts at some position and has to deliver items to various locations. However, every item has an expiration date and is useful only if it can be delivered to its destination by this date. A natural goal for the delivery-person is to deliver as many items as possible within their deadlines, or more generally to maximize his profit, if each item has a value associated with it. We call this problem the *Deadline-TSP*. Note that if the deadlines of all the nodes are the same, the problem reduces to the well known *Orienteering* problem [AMN98, AABV99, GLV87], for which the first constant factor approximation for this problem was recently given [BCK⁺03].

A generalization of the Deadline-TSP, the *Time Window* problem, allows nodes to have release times in addition to deadlines. The traveling salesperson is allowed to visit a node only in the time window defined by the release times and deadlines. This problem has been studied extensively in the OR literature under the name *Vehicle Routing Problem with Time Windows*. Various heuristics [DDS92, Sav85, Tha95, TLZ00], such as local search, Simulated Annealing and Genetic algorithms, as well as cutting plane and branch and bound methods [TOVS93, KKT87, KR92] have been studied for solving this problem optimally. Optimal algorithms for stochastic inputs have also been proposed.

In approximation algorithms literature, people have studied the geometric version of the time-window problem. Several constant factor approximations [BYES03, Tsi92, KN01] have been proposed for the case of points on a line. Recently, Chekuri and Kumar [CK04] considered the time-window problem on a general graph, but with a constant number of different deadlines. They give a constant factor approximation for this problem. Our paper is the first to give an approximation algorithm for the general case of the Deadline-TSP with arbitrary deadlines.

*Department of Computer Science, Carnegie Mellon University. {nikhil,avrim,shuchi}@cs.cmu.edu.

†University of California, Los Angeles. awm@cs.ucla.edu.

Another motivation for our problem comes from a classic scheduling problem known as scheduling with sequence dependent setup times [AGA99, YL99, WS95, BD78]. In this problem we are given several jobs released at time zero, each having a production duration p_j and a delivery date D_j . In addition, for each pair of jobs, there is a setup time s_{ij} , which is incurred when job j is undertaken following job i . The goal is to schedule jobs as efficiently as possible on a single machine. This can be modeled as a graph problem by assigning a length of s_{ij} to the edge between i and j . Production durations can also be incorporated by adding $p_j/2$ to every edge incident on the node j , and subtracting the same from the deadline D_j .

The scheduling setting is fairly general and models many problems. For example, one objective may be to minimize the makespan, which is equivalent to the TSP problem. Likewise, the objective of completing as many jobs as possible by their delivery date is equivalent to Deadline-TSP. Interestingly, meeting target delivery dates has been declared as the most important scheduling objective for production planners by Wisner and Siferd [WS95]. Several heuristics [Jor98, GPG00] have been proposed for this (and other related problems), however, no approximations were known prior to our work.

We give an $O(\log n)$ approximation for the Deadline-TSP, based on a 3-approximation that we provide for an extension of the Orienteering problem, that we call “point-to-point orienteering”. In point-to-point orienteering, both the start node and end node of the path are given, and the goal is to travel from the start to the end, traveling distance at most D , and visiting (approximately) as many points as possible. Note that we approximate the number of points, and not the distance traveled. Having an algorithm for the point-to-point problem is useful for Deadline-TSP because it means that if we can break up the problem into several subparts, then we can attack each piece individually and patch the solutions together without worrying about spending too much in connecting the different parts. Since point-to-point orienteering generalizes the standard version of this problem, this improves on the previous 4-approximation of [BCK⁺03].

We also give an algorithm for the Deadline-TSP, that achieves a bicriteria optimization — it achieves a constant factor approximation to the reward, while exceeding the deadlines by a small factor. In particular, for any arbitrary $\epsilon > 0$, our algorithm collects an $O(\log 1/\epsilon)$ fraction of the reward, while exceeding deadlines by a factor of $1 + \epsilon$.

The rest of this paper is organized as follows. We begin with some notation and definitions in Section 2. In Section 3, we give a 3-approximation algorithm for the point-to-point orienteering problem. Section 4 contains some constant factor approximations for special cases of Deadline-TSP, that we then use to achieve our bicriteria approximation. Section 5 contains an $O(\log n)$ -approximation algorithm. We conclude in Section 6.

2 Notation and Preliminaries

Let $G = (V, E)$ be a weighted graph, with a start node r , a prize (profit) function $\Pi : V \rightarrow \mathcal{Z}^+$, deadlines $D : V \rightarrow \mathcal{Z}^+$, and a length function $\ell : E \rightarrow \mathcal{Z}^+$.

For any two nodes u and v , let $\ell(u, v)$ denote the shortest distance between u and v . Given a path P from u to v , let $t_P(u, v)$ denote the time taken (distance) to reach v from u along the path P . The excess along the path P is defined as $\epsilon_P(u, v) = t_P(u, v) - \ell(u, v)$. We abbreviate the time taken by a path P rooted at r to reach a node v , $t_P(r, v)$, by $t_P(v)$. We denote the optimal path by \mathcal{O} . Let \mathcal{O} also denote the set of nodes visited by the optimal path before their deadlines. A path starting at root r collects the prize $\Pi(v)$ at node v , if it reaches v before its deadline $D(v)$. For a path P and set $S \subseteq V$, let $\Pi_P(S) = \sum_{v \in S: t_P(v) \leq D(v)} \Pi(v)$ and $\Pi_P = \Pi_P(V)$. Note that a path can visit a node multiple times, but the prize at any node can be collected at most once.

The goal in the Deadline-TSP is to construct a path starting at r , that maximizes the total prize.

The $u - v$ Orienteering problem is a special case of this problem in which all nodes have the same deadline, $D(x) = D$, $\forall x \in V$, and any path must start from the node u and end at the node v .

The minimum excess path problem is defined as follows. Given a graph with nodes u and v , and a prize quota k , the problem is to find a path P from u to v that collects at least k prize, and has the minimum possible excess $\epsilon_P(u, v)$. Note that in terms of exact solutions, this problem is the same as the k -TSP problem, in which the goal is to find the shortest path between u and v that collects at least k prize, but we are subtracting $\ell(u, v)$ from the objective, so the min-excess problem is more difficult in terms of approximation. Blum et al [BCK⁺03] give a $(2 + \epsilon)$ -approximation for the excess problem (if the optimum path is \mathcal{O} , the algorithm finds a path that collects a prize at least k and has length at most $d(u, v) + (2 + \epsilon)\epsilon_{\mathcal{O}}(u, v)$), and then use this to get a 4-approximation to the orienteering problem.

We begin by showing how we can use the $(2 + \epsilon)$ -approximation to the minimum-excess path problem to find a path from u to v that has length at most $t_{\mathcal{O}}(u, v)$, and collects prize at least $k/3$. In other words, this gives a 3-approximation to the $u - v$ Orienteering problem, improving over [BCK⁺03] (in fact, this is stronger than the version of the problem solved in [BCK⁺03] because we get to specify the ending point of the path, and not just the starting point).

We then show how we can use this as a subroutine to obtain a bicriteria approximation—for any $\epsilon > 0$, our algorithm outputs a path P with $\Pi_P^\epsilon \geq \Omega(\frac{1}{\log(1/\epsilon)})\Pi_{\mathcal{O}}$, where $\Pi_P^\epsilon = \sum_{v: t_P(v) \leq (1+\epsilon)D(v)} \Pi(v)$. That is, for a constant ϵ , our algorithm obtains a constant fraction of the reward, while exceeding deadlines by a $1 + \epsilon$ factor. The same algorithm also gives us an $O(\log D_{\max})$ approximation without approximating deadlines, where $D_{\max} = \max_{v \in V} D(v)$ is the maximum deadline in the graph.

Finally, using $u - v$ Orienteering, we also give a $3 \log n$ approximation for the Deadline-TSP, which outperforms the earlier approximation when D_{\max} is large.

For any path P , the restriction of the path P to a set S of vertices, $P|_S$, denotes the path that visits nodes of S in the order that P visited them, and does not visit any other nodes. Note that, for all S and P , $t_{P|_S}(v) \leq t_P(v)$ for all $v \in S$.

3 Point-to-point Orienteering

In this section we show how we can use a $(2 + \epsilon)$ -approximation algorithm for the minimum excess problem, to achieve a 3-approximation to the point-to-point orienteering problem.

We begin with some properties of excess. Let P be any $u - v$ path that visits nodes in the order $u_0 = u, u_1, \dots, u_l = v$.

Fact 1 $\epsilon_P(u_0, u_i)$ is increasing in i .

Fact 2 The excess function is sub-additive. That is, for any nodes u_i, u_j, u_k with $0 \leq i < j < k \leq l$, $\epsilon_P(u_i, u_j) + \epsilon_P(u_j, u_k) \leq \epsilon_P(u_i, u_k)$.

Proof: The first fact follows from the triangle inequality. The second follows by rewriting $\epsilon_P(u_i, u_j) + \epsilon_P(u_j, u_k)$ as $t_P(u_i, u_j) - \ell(u_i, u_j) + t_P(u_j, u_k) - \ell(u_j, u_k) = t_P(u_i, u_k) - \ell(u_i, u_j) - \ell(u_j, u_k)$ and observing that $\ell(u_i, u_j) + \ell(u_j, u_k) \geq \ell(u_i, u_k)$ by the triangle inequality. ■

The algorithm *P2P* is given in Figure 1. Note that by design, the algorithm produces a path of length at most D . We just need to show that it visits enough points.

Theorem 1 *The algorithm P2P is a 3-approximation to the point-to-point orienteering problem.*

Proof: Consider the optimum path \mathcal{O} from u to v . Break this path into three pieces, each having at least $1/3$ of the total prize value, and let x and y be the endpoints of the portion such that $\epsilon_{\mathcal{O}}(x, y)$

Input: Graph $G = (V, E)$; special nodes u and v ; length bound D .
Output: Path P with $\Pi_P \geq \frac{1}{3}\Pi_{\mathcal{O}}$ and length at most D .

1. For every pair of nodes x and y , we will consider a path which proceeds from u to x , then visits some vertices while traveling from x to y , and then travels directly from y to v . The allowed excess of the path from x to y is $\epsilon_{xy} = D - \ell(u, x) - \ell(x, y) - \ell(y, v)$. Using the $2 + \epsilon$ -approximation to the minimum excess problem from [BCK⁺03], we compute a path from x to y of excess at most ϵ_{xy} that visits at least as many vertices as the best path from x to y with excess $\epsilon_{xy}/3$.
 2. We now pick the pair (x, y) that maximizes the total reward collected on the computed path. We then travel from u to x via a line, then x to y via the chosen path, then y to v .
-

Figure 1: Algorithm $P2P$ — 3-approximation for $u - v$ Orienteering

is smallest. Consider now the path O' that travels directly from u to x , then follows \mathcal{O} to y , and then travels directly from y to v . Because we chose x, y such that $\epsilon_{\mathcal{O}}(x, y)$ is the smallest of the three segments, and by the triangle inequality, it must be that by traveling along O' rather than \mathcal{O} , we save a total of at least $2\epsilon_{\mathcal{O}}(x, y)$; that is, $t_{\mathcal{O}}(u, v) - t_{O'}(u, v) \geq 2\epsilon_{\mathcal{O}}(x, y)$. This is enough to pay for the added length produced by applying the min-excess approximation from x to y . Formally, by definition, $\epsilon_{xy} = t_{\mathcal{O}}(u, v) - t_{O'}(x, y) + \epsilon_{\mathcal{O}}(x, y)$, and plugging in the above inequality, this is at least $3\epsilon_{\mathcal{O}}(x, y)$. Since the min-excess algorithm gets at least as much prize value as the best possible path from x to y with excess $\epsilon_{xy}/3$, it is guaranteed to get at least as much as this portion of the optimal path, which is at least $1/3$ of the total prize of the optimal path. ■

4 A Bicriteria Approximation

In this section, we describe an algorithm that for any $\epsilon > 0$, obtains an $O(\log \frac{1}{\epsilon})$ fraction of the reward obtained by \mathcal{O} , if it is allowed to exceed deadlines by a factor of $1 + \epsilon$.

We begin with a few special purpose algorithms, and then describe how to combine them for the general case. We first consider the case when \mathcal{O} visits a large fraction of the nodes very close to their deadlines (the *small margin* case). Then we consider the case when \mathcal{O} visits most nodes much before their deadlines (the *large margin* case). Towards the end of this section, we show that the bicriteria result also implies a $O(\log D_{\max})$ -approximation to Deadline-TSP.

4.1 The small margin case

At the heart of our bicriteria approximation is a procedure that obtains a constant fraction of the optimal reward (while approximating deadlines to within a small factor), if \mathcal{O} visits most nodes $v \in \mathcal{O}$ very close to their deadlines (within some multiplicative factor).

Let ϵ be a fixed constant. Consider the set of nodes $V_{\epsilon} = \{v : D(v)/(1 + \epsilon) \leq t_{\mathcal{O}}(v) \leq D(v)\}$. The algorithm \mathcal{A} will obtain a constant fraction of $\Pi_{\mathcal{O}}(V_{\epsilon})$ as reward, while approximating deadlines to within a factor of $1 + \epsilon$.

Let $f = \frac{1}{1+\epsilon}$. We divide the nodes of V_{ϵ} into segments as follows. Segment S_j consists of nodes in V_{ϵ} that have deadlines in $(f^j D_{\max}, f^{j-1} D_{\max}]$ (Figure 2). The following Lemma 2 shows that for any j , all vertices in $S_j \cap \mathcal{O}$ are visited after all vertices in $S_{j+2} \cap \mathcal{O}$ (see Figure 2). Furthermore, if we use point-to-point orienteering to approximate the path in segment S_j , then every point in this segment is visited before the last time that \mathcal{O} visits this segment. Lemma 3 shows that this time is at most $(1 + \epsilon)$ times the deadline of the any node in S_j . Thus the path returned by path-to-path orienteering exceeds deadlines by a factor of at most $1 + \epsilon$.

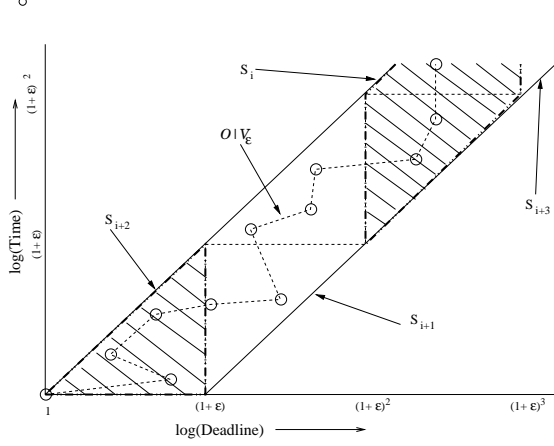


Figure 2: Division of set V_ϵ into segments S_j

Lemma 2 For any j and any nodes $u \in S_j \cap \mathcal{O}$ and $v \in S_{j+2} \cap \mathcal{O}$, $t_{\mathcal{O}}(v) < t_{\mathcal{O}}(u)$.

Proof: We have $D(v) \leq f^{j+1}D_{\max}$. So, $t_{\mathcal{O}}(v) \leq D(v) \leq f^{j+1}D_{\max}$. Likewise, $D_u > f^j D_{\max}$. So, by the definition of V_ϵ , $t_{\mathcal{O}}(u) \geq f D_u > f^{j+1}D_{\max}$. ■

Lemma 3 For any two nodes $u, v \in S_j \cap \mathcal{O}$, $t_{\mathcal{O}}(u) < (1 + \epsilon)D(v)$.

Proof: By the definition of V_ϵ , $t_{\mathcal{O}}(u) \leq D_u$. But, since u and v belong to the same segment, $D_u \leq f^{j-1}D_{\max} < D(v)/f$. Thus, $t_{\mathcal{O}}(u) < D(v)/f$. ■

The two lemmas suggest a natural strategy for approximating the reward collected by \mathcal{O} in V_ϵ . Let $\mathcal{S}_0 = \cup_{j=0}^{\infty} S_{2j}$ and $\mathcal{S}_1 = \cup_{j=0}^{\infty} S_{2j+1}$. Then, $\mathcal{S}_0 \cup \mathcal{S}_1 = V_\epsilon$. Then, one of the two sets contains at least half of the total reward in V_ϵ . Let this be \mathcal{S}_k .

We approximate the reward obtained by $\mathcal{O}' = \mathcal{O}|_{\mathcal{S}_k}$ as follows. Guess the first and the last vertex that \mathcal{O}' visits in S_j , and the time at which it visits them. Construct a path of the guessed length between the two guessed end points using the *P2P* algorithm, and patch up the paths to get path P .

From Lemma 3, we know that the path P visits nodes in \mathcal{S}_k before $(1 + \epsilon)$ times their deadlines. We lose a factor of 2 in reward by leaving out vertices in \mathcal{S}_{1-k} , and another factor of 3 by using algorithm *P2P*. Thus, we get the following theorem. Figure 3 describes the algorithm in detail.

Theorem 4 Algorithm \mathcal{A} returns a path P with $\Pi_P^\epsilon \geq \frac{1}{6}\Pi_{\mathcal{O}}(V_\epsilon)$.

4.2 The large margin case

Let $V_{\frac{1}{2}} = \{v : t_{\mathcal{O}}(v) \leq \frac{D(v)}{2}\}$. In this section we will describe an algorithm that collects a constant fraction of the reward $\Pi_{\mathcal{O}}(V_{\frac{1}{2}})$.

For all nodes v , define new deadlines $D'(v) = \frac{D(v)}{2}$. Note that for all $v \in V_{\frac{1}{2}}$, \mathcal{O} visits v before its new deadline $D'(v)$. We divide nodes into subsets as follows. For $i \geq 0$, the set \mathcal{S}_i contains nodes v that have $D'(v) \in [(1.5)^i, (1.5)^{i+1})$. Let $\beta = 5$. For $j \in \{0, \dots, \beta - 1\}$, define $\mathcal{S}_j = \cup_{i \geq 0} \mathcal{S}_{\beta i + j}$. Then, $\cup_{j \leq \beta-1} \mathcal{S}_j = V$.

Input: Graph $G = (V, E)$ with deadlines $D(v)$; Constant ϵ ; $f = \frac{1}{1+\epsilon}$.

Output: Path P with $\Pi_P \geq \frac{1}{6}\Pi_{\mathcal{O}}(V_\epsilon)$ and $t_P(v) \leq (1+\epsilon)D(v)$ for all $v \in P$, where V_ϵ is defined as above.

1. For all j , for all $u, v \in S_j$, and for all integers $t \leq f^{j-1}D_{\max}$, use algorithm $P2P$ to find a path of length t from u to v . Let this reward be $\pi(u, v, t)$.
2. For $k = 0, 1$ do the following:
 - (a) Let $\pi'(v, j, t)$ denote the (approximately) maximum reward that can be collected from S_k up to segment $S_j \subset S_k$ using a path of length t that ends at $v \in S_j$.
 - (b) Compute $\pi'(v, j, t)$ using the following recurrence.

$$\pi'(v, j, t) = \max_{t' \leq f^{j-1}D_{\max}, u \in S_j, u' \in S_{j+2}} \{\pi'(u', j-2, t') + \pi(u, v, t - t' - \ell(u', u))\}$$

3. Output the path corresponding to the maximum prize collected until the last segment.
-

Figure 3: Algorithm \mathcal{A} —Bicriteria approximation for the small margin case

Let P_i be a path returned by the $P2P$ algorithm with parameter $D = (1.5)^{i+1}$ when applied to the graph induced by $\{r\} \cup S_i$. Note that P_i collects at least $\frac{1}{3}\Pi_{\mathcal{O}}(S_i)$ reward. Let T_i be a tour that starts at the root, follows path P_i and returns to the root. For some $j < \beta$, consider the path constructed by appending all $T_{\beta i+j}$ for $i \geq 0$. Let this path be called Q_j .

Lemma 5 For any i and j and $v \in S_{\beta i+j}$, $t_{Q_j}(v) \leq D(v)$.

Proof: The length of the path Q_j upto set $S_{\beta i+j}$ is $\sum_{k < i} |T_{\beta k+j}| + |P_{\beta i+j}| \leq \sum_{k < i} 2|P_{\beta k+j}| + |P_{\beta i+j}| \leq 2 \sum_{k < i} (1.5)^{\beta k+j+1} + (1.5)^{\beta i+j+1} = 2(1.5)^{j+1} \frac{(1.5)^{\beta i} - 1}{(1.5)^{\beta} - 1} + (1.5)^{\beta i+j+1} \leq (1.5)^{\beta i+j+1} (\frac{2}{1.5^{\beta}-1} + 1)$. On the other hand, the deadline of v is $D(v) = 2D'(v) \geq 2(1.5)^{\beta i+j}$. Thus, $t_{Q_j}(v) \leq \frac{1.5}{2} (\frac{2}{1.5^{\beta}-1} + 1) D(v)$. Taking $\beta = 5$, we get the result. \blacksquare

Note that the paths Q_j for $j \in \{0, \dots, \beta - 1\}$ together cover all the nodes in $V_{\frac{1}{2}}$. Furthermore, we have $\Pi_{Q_j} \geq \frac{1}{3} \sum_i \Pi_{\mathcal{O}}(S_{\beta i+j})$. Thus, one of the paths Q_j gives a $3\beta = 15$ approximation to the reward collected by \mathcal{O} in $V_{\frac{1}{2}}$. Our algorithm for finding the best Q_j is given in Figure 4.

Theorem 6 Algorithm \mathcal{B} returns a path P with $\Pi_P \geq \frac{1}{15}\Pi_{\mathcal{O}}(V_{\frac{1}{2}})$.

Input: Graph $G = (V, E)$ with deadlines $D(v)$.

Output: Path Q with $\Pi_Q \geq \frac{1}{15}\Pi_{\mathcal{O}}(V_{\frac{1}{2}})$ and $t_Q(v) \leq D(v)$ for all $v \in Q$, where $V_{\frac{1}{2}}$ is defined as above.

1. For all i , use the algorithm $P2P$ on graph $G(\{r\} \cup S_i)$ to construct a path P_i with length at most $(1.5)^{i+1}$. Let T_i be the corresponding tour.
 2. For all $j \in \{0, \dots, \beta - 1\}$, let Q_j be the concatenation of $P_{\beta i+j}$ for all $i \geq 0$ and let $\pi_j = \sum_i \Pi_{P_{\beta i+j}}(S_{\beta i+j})$.
 3. Return the path Q_j corresponding to the maximum reward π_j over all j .
-

Figure 4: Algorithm \mathcal{B} —Approximation for the large margin case

4.3 The general case

Now we will give an algorithm that produces a bicriteria approximation for the entire graph. In particular, given a parameter ϵ , we will construct a path that obtains reward at least $O(\log \frac{1}{\epsilon})\Pi_{\mathcal{O}}$ from nodes reached within a factor of $1 + \epsilon$ of their deadlines. As before, let $f = \frac{1}{1+\epsilon}$. Let s be defined as the smallest integer for which $f^{2^s} \leq \frac{1}{2}$. Then $s = O(\log \frac{1}{\epsilon})$.

Our algorithm proceeds as follows. Divide all the nodes into $s + 2$ groups as follows. Group i , $1 \leq i \leq s$, is given by $V_i = \{v : t_{\mathcal{O}}(v) \in (f^{2^i} D(v), f^{2^{i-1}} D(v))\}$. Group 0 is given by $V_0 = \{v : t_{\mathcal{O}}(v) \in (f D(v), D(v))\}$. Group $s + 1$ is defined as $V_{s+1} = \{v : t_{\mathcal{O}}(v) \in (0, D(v)/2]\}$. These groups together cover all the nodes in \mathcal{O} . So, one of the groups contains at least a $\frac{1}{s+2}$ fraction of the total reward collected by \mathcal{O} . Let V_i be such a group.

If $i = 0$, we can apply algorithm \mathcal{A} right away and obtain a path P with $\Pi_P \geq \frac{1}{6}\Pi_{\mathcal{O}}(V_0)$ that visits its nodes within a factor of $(1 + \epsilon)$ of their deadlines.

Consider the case when $1 \leq i \leq s$. Scale all the deadlines down by a factor of $f^{2^{i-1}}$, that is, define $D'(v) = f^{2^{i-1}} D(v)$. Then the path \mathcal{O} visits all nodes in V_i at time $t_{\mathcal{O}}(v) \in (f^{2^{i-1}} D'(v), D'(v))$. Now apply algorithm \mathcal{A} with parameter $f^{2^{i-1}}$. Then, the obtained path collects reward $\pi_P \geq \frac{1}{6}\Pi_{\mathcal{O}}(V_i)$ and visits all nodes v before time $D'(v)(1 + \epsilon)^{2^{i-1}} = D(v)$.

Finally consider the case when $i = s + 1$. Note that this is the large margin case considered in a previous subsection. So we can use algorithm \mathcal{B} to obtain a 15-approximation in this case.

Putting everything together, we get the following theorem. The algorithm is described in Figure 5.

Theorem 7 *Algorithm \mathcal{C} returns a path P with $\Pi_P^\epsilon \geq \frac{1}{15(s+2)}\Pi_{\mathcal{O}} = \Omega(\frac{1}{\log \frac{1}{\epsilon}})\Pi_{\mathcal{O}}$.*

Input: Graph $G = (V, E)$ with deadlines $D(v)$; parameter ϵ .

Output: Path P with $\Pi_P \geq \Omega(\frac{1}{\log \frac{1}{\epsilon}})\Pi_{\mathcal{O}}$ and $t_P(v) \leq (1 + \epsilon)D(v)$ for all $v \in P$.

1. Let $f = \frac{1}{1+\epsilon}$ and s be the smallest integer for which $f^{2^s} \leq \frac{1}{2}$.
 2. Apply algorithm \mathcal{A} with parameter f to the graph, and let P_0 be the path obtained.
 3. Apply algorithm \mathcal{B} to the graph, and let P_{s+1} be the path obtained.
 4. For all $i \in \{1, \dots, s\}$, do the following:
 - (a) For all $v \in V$, define $D'(v) = D(v)f^{2^{i-1}}$.
 - (b) Apply algorithm \mathcal{A} with parameter $f^{2^{i-1}}$ to the graph with the new deadlines D' , and let P_i be the path obtained.
 5. Among the paths constructed above, return the one with the maximum reward.
-

Figure 5: Algorithm \mathcal{C} —Bicriteria approximation for the general case

As a simple corollary of the above theorem, we get an $O(\log D_{\max})$ -approximation to Deadline-TSP without exceeding deadlines.

Corollary 8 *Algorithm \mathcal{C} with $\epsilon = 1/D_{\max}$ gives an $O(\log D_{\max})$ -approximation to the Deadline-TSP.*

Proof: Note that from Theorem 7, we know that the path P collects an $O(\log D_{\max})$ fraction of the reward from nodes v visited before $(1 + \epsilon)D(v)$. For $\epsilon = 1/D_{\max}$, $(1 + \epsilon)D(v) \leq D(v) + 1$. Since, all edge lengths and deadlines are integral by assumption, node v is visited by $D(v)$. ■

5 An $O(\log n)$ Approximation

In this section we give an $O(\log n)$ approximation algorithm for the Deadline-TSP problem. We begin with some notation.

Let $r = u_0, u_1, \dots, u_l$ denote the vertices visited by the optimal path \mathcal{O} . It is convenient to view the vertices in \mathcal{O} as lying on the two dimensional plane, with the horizontal and vertical axes corresponding to time and deadlines respectively. That is, vertex u_i lies at the point $p_i = (t_{\mathcal{O}}(u_i), D(u_i))$ (See Figure 6(a)).

We call a vertex u_i *minimal* if for any other vertex u_j , either $t_{\mathcal{O}}(u_j) \leq t_{\mathcal{O}}(u_i)$ or $D(u_j) \geq D(u_i)$. Pictorially, these are vertices that form the lower envelope on the points p_i (see Figure 6). Let $\mathcal{M} = \{v'_0, \dots, v'_m\}$ denote the set of minimal vertices, ordered in increasing order of deadlines. Without loss of generality, we assume that $D(r) = 0$, so $r \in \mathcal{M}$. Similarly, for the purposes of analysis we assume that there is a dummy vertex u_{end} which is at distance $2 \sum_{e \in E} \ell(e)$ from the root and has a deadline of $4 \sum_{e \in E} \ell(e)$. Without loss of generality we can assume that u_{end} is always visited in the end by any tour. Note that $u_{end} \in \mathcal{M}$.

For any three minimal vertices $v'_h, v'_j, v'_k \in \mathcal{M}$ with $h \leq j \leq k$, let $\mathcal{R}(h, j, k)$ denote the set of vertices u in \mathcal{O} such that $D(v'_j) \leq D(u) \leq D(v'_k)$ and $t_{\mathcal{O}}(v'_h) \leq t_{\mathcal{O}}(u) \leq t_{\mathcal{O}}(v'_j)$. Thus, $\mathcal{R}(h, j, k)$ is the set of points lying in the rectangular region defined by the constraints above (see Figure 6). An interesting property of a rectangle is that we consider the optimum path \mathcal{O} restricted to the vertices within a rectangle, then all the vertices are visited no later than the vertex with the least deadline. In a sense, this will allow to apply to the orienteering subroutine later.

Two rectangles $\mathcal{R}_1 = \mathcal{R}(h_1, j_1, k_1)$ and $\mathcal{R}_2 = \mathcal{R}(h_2, j_2, k_2)$ are called *disjoint* if $h_2 \geq j_1$ and $j_2 \geq k_1$, (or equivalently if $h_1 \geq j_2$ and $j_1 \geq k_2$). Pictorially, \mathcal{R}_1 and \mathcal{R}_2 are disjoint if for any two points r_1 in the interior of \mathcal{R}_1 and r_2 in the interior of \mathcal{R}_2 , both the x -coordinates and the y -coordinates of r_1 and r_2 are different (Figure 6). Observe that no vertex can lie in two disjoint rectangles. Finally, a collection of rectangles $\mathcal{C} = \{\mathcal{R}_1, \dots, \mathcal{R}_r\}$ is called disjoint if for all pairs $\mathcal{R}_i, \mathcal{R}_j \in \mathcal{C}$, \mathcal{R}_i and \mathcal{R}_j are disjoint. This notion of disjointness of rectangles will be useful as it will allow us to solve an orienteering problem separately for various rectangles, and patch up the paths obtained without double counting the reward.

The main idea of our $O(\log n)$ approximation is the following. First, we show that there is a $\log n$ size family of disjoint collections of rectangles $\{\mathcal{C}_1, \dots, \mathcal{C}_{\log n}\}$ such that each vertex $u_i \in \mathcal{O}$ lies in a rectangle contained in at least one \mathcal{C}_i . This implies that the total reward contained in this family of collections is at least $\Pi_{\mathcal{O}}$. Therefore, there is some collection \mathcal{C}_i that has at least a $1/\log n$ fraction of this reward. Second, we will give a polynomial time procedure to compute a path that collects at least an $O(1)$ fraction of the reward contained in the best disjoint collection of rectangles. Together, these will imply an $O(\log n)$ approximation.

We first describe the family $\mathcal{C}_1, \dots, \mathcal{C}_{\log n}$. The collections \mathcal{C}_i are parameterized by sets of integers. Let S be a set of integers $\{n_1, \dots, n_s\}$, where each $n_i \leq m$, and, let $n_0 = 0$ and $n_{s+1} = m$. We can associate a disjoint collection of rectangles with this set, $\mathcal{C}(S) = \{\mathcal{R}(n_{i-1}, n_i, n_{i+1}) : 1 \leq i \leq s\}$. For $i \in \{0, \dots, \log m - 1\}$, let $S_i = \{j2^i + 2^{i-1} : 0 \leq j \leq 2^{\log m - i} - 1\}$, and let $\mathcal{C}_i = \mathcal{C}(S_i)$ (Figure 6). Thus we have a family $\mathcal{F} = \{\mathcal{C}_1, \dots, \mathcal{C}_{\log m}\}$ consisting of $\log m \leq \log n$ collections.

Lemma 9 *For each vertex u visited in the optimum tour \mathcal{O} , there is at least one collection $\mathcal{C}_i \in \mathcal{F}$, such that u lies in some rectangle in \mathcal{C}_i .*

Proof: Consider a vertex $u \in \mathcal{O}$ and let v'_i be the minimal vertex for which $D(v'_i) \leq D(u) < D(v'_{i+1})$. Likewise, let v'_j be the minimal vertex such that $t_{\mathcal{O}}(v'_{j-1}) < t_{\mathcal{O}}(u) \leq t_{\mathcal{O}}(v'_j)$. Observe that $i \geq j$. If $i = j$, then u is the minimal vertex v'_i and in this case, $u \in \mathcal{C}_0$. If $i \neq j$, observe that i and j lie in some rectangle in \mathcal{C}_b if and only if $|S_b \cap [i, j]| = 1$, i.e. S_b contains exactly one number between i and j . Let

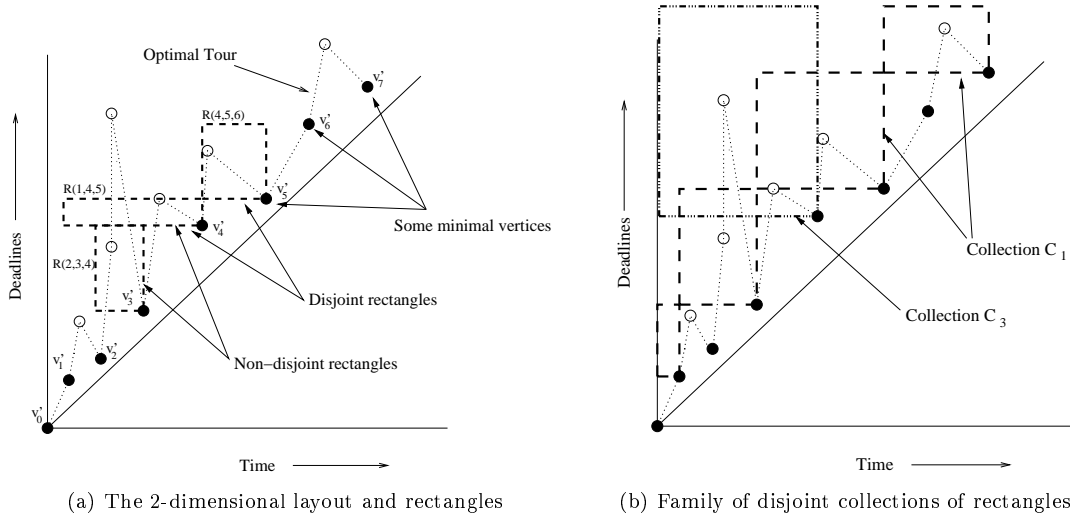


Figure 6: An example illustrating the construction of the rectangles $R(h, j, k)$.

b be such that $2^b \leq i - j < 2^{b+1}$. For this choice of b , either $|S_b \cap [i, j]| = 1$ or $|S_b \cap [i, j]| = 2$. In the first case, we are already done. In the second case, let x and y be the points in $S_b \cap [i, j]$, then observe that $(x + y)/2 \in S_{b+1}$ and since $i - j < 2^{b+1}$, exactly one point from S_{b+1} lies in the range $[i, j]$. ■

5.1 Approximating the reward in the best collection

We now show how we can find a path such that the total reward collected in that path is a constant factor of the reward in the best collection.

The idea is the following: Consider the collection $\mathcal{C} \in \mathcal{F}$ that has the maximum reward. Let us consider the optimum path restricted to the vertices in \mathcal{C} . Then this path remains a feasible path (i.e. meets all its deadlines) for a more restricted instance, where for each vertex v in $\mathcal{R}(i, j, k) \in \mathcal{C}$ we assign a smaller deadline $D'(v) = D(v') \leq D(v)$. Thus, if we could run the point to point orienteering subroutine on the instance restricted to vertices in $\mathcal{R}(i, j, k)$, we would be guaranteed a constant fraction of the reward that \mathcal{O} collects in the rectangle $\mathcal{R}(i, j, k)$. However, we do not know which vertices are minimal, and therefore do not know which vertices lie in $\mathcal{R}(i, j, k)$ (the definition of a minimal vertex depends on when \mathcal{O} visits that vertex). To get around this problem, we can do the following: Observe that the vertices v'_j in \mathcal{M} have increasing deadlines. This allows to write dynamic program in a natural way. We can arrange all the vertices of G in the increasing order of deadlines as v_1, \dots, v_n . For every $1 \leq j \leq k \leq n$, we consider the graph $G_{j,k}$ restricted to vertices v_l such that $j \leq l < k$, and solve a point to point orienteering instance (assuming some start time to account for distance traveled before considering this interval) such that each vertex is visited before the deadline of v'_j . We now give the details.

Lemma 10 *We can compute in time $O(\text{poly}(n, D_{\max}))$ a feasible path that collects at least a third of the the reward collected by the best collection \mathcal{C}_i .*

Proof: Let v_1, \dots, v_n denote the vertices in G in the increasing order of their deadlines. We compute the following quantity: For every tuple of vertices v_j, v_k such that $j \leq k$, and every vertex v_g and v_h such that $D(v_g), D(v_h) \in [D(v_j), D(v_k)]$ and start and finish times t_1 and t_2 , such that $t_1 \leq t_2 \leq D(v_j)$, let $\pi(j, k, g, h, t_1, t_2)$ be the reward collected by the optimum path that begins at v_g at time t_1 , finishes

at v_h at time t_2 and only visits vertices such that their deadlines are in the interval $[D(v_j), D(v_k))$. To compute this approximately (upto a factor of 3), we run the point to point orienteering subroutine on the graph restricted to nodes v such that for $D(v_j) \leq D(v) < D(v_k)$.

Having obtained these $O(n^4 D^2)$ quantities, we use a dynamic program to find the maximum reward path obtained by patching the paths corresponding to disjoint intervals $[v_j, v_k]$, computed above. Note that since the vertices are ordered by deadlines, no vertex is double counted in the reward. Moreover the path obtained is feasible, since every vertex v in an interval $[v_j, v_k]$ is visited before $D(v_j)$ and hence before $D(v)$. Finally, observe that any path corresponding to a collection \mathcal{C}_i would be considered by this dynamic program, as this corresponds to patching intervals corresponding to the disjoint rectangles in \mathcal{C}_i . Hence modulo the factor 3 that we lose in the point to point orienteering subroutine, we can obtain at least the reward contained in the optimum collection \mathcal{C}_i . ■

By Lemma 9 and 10 we have that,

Theorem 11 *The algorithm is described in Figure 7 is an $O(\log n)$ approximation algorithm for the Deadline-TSP problem.*

Input: Graph $G = (V, E)$ with deadlines $D(v)$.

Output: Path P with $\Pi_P \geq \frac{1}{3 \log n} \Pi_{\mathcal{O}}$.

1. Let V_{jk} denote the set of vertices with deadlines between $D(v_j)$ and $D(v_k)$. For all $j \neq k \leq n$, for all $t \leq D(v_j)$, and for all $v_g, v_h \in V_{j,k}$, apply algorithm *P2P* to the graph restricted to V_{jk} with distance bound t , and let $\pi(j, k, g, h, t)$ denote the reward obtained.
2. Let $\pi'(k, t)$ with $t \leq D(v_k)$ denote the (approximately) maximum reward that can be obtained by a path of length t visiting nodes with deadline at most $D(v_k)$.
3. For all $k \leq n$ and $t \leq D(v_k)$, compute $\pi'(k, t)$ and the corresponding path by the following recurrence

$$\pi'(k, t) = \max_{j \leq g, h \leq k, \text{with } D(v_j) \geq t, t' \leq t} \{\pi'(j, t') + \pi(j, k, g, h, t - t')\}$$

4. Return the path corresponding to maximum reward computed in the last step.
-

Figure 7: $O(\log n)$ -approximation algorithm

6 Conclusion

We present a log-factor approximation algorithm for the Deadline-TSP problem based on an approximation to the point-to-point orienteering problem. We also give an improved 3-approximation to the point-to-point orienteering problem. The main open question is whether it is possible to achieve a constant-factor approximation for the Deadline-TSP. Interestingly, obtaining a constant factor approximation to the unrooted version of the problem (where the path can begin at any node) is as hard as for the rooted version. Another open problem would be to consider arbitrary release times for vertices. There is no approximation algorithm known for this case in general graphs.

Acknowledgements

This work was supported in part by NSF grants CCR-0105488 and NSF-ITR CCR-0122581.

References

- [AABV99] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala. Improved approximation guarantees for minimum-weight k -trees and prize-collecting salesmen. *Siam J. Computing*, 28(1):254–262, 1999.
- [AGA99] A. Allahverdi, J. Gupta, and T. Aldowaisan. A review of scheduling research involving setup considerations. *Omega, Int. Journal of Management Science*, 27:219–239, 1999.
- [AMN98] E. M. Arkin, J. S. B. Mitchell, and G. Narasimhan. Resource-constrained geometric network optimization. In *Symposium on Computational Geometry*, pages 307–316, 1998.
- [BCK⁺03] A. Blum, S. Chawla, D. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. In *Proceedings of the 44th Foundations of Computer Science*, 2003.
- [BD78] J. Bruno and P. Downey. Complexity of task sequencing with deadlines, set-up times and changeover costs. *SIAM Journal on Computing*, 7(4):393–404, 1978.
- [BYTES03] R. Bar-Yehuda, G. Even, and S. Shahar. On approximating a geometric prize-collecting traveling salesman. In *Proceedings of the European Symposium on Algorithms*, 2003.
- [CK04] C. Chekuri and A. Kumar. Maximum coverage problem with group budget constraints and applications, 2004.
- [DDS92] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1992.
- [GLV87] B.L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34:307–318, 1987.
- [GPG00] M. Gravel, W. Price, and C. Gagn. Scheduling jobs in a alcan aluminium factory using a genetic algorithm. *International Journal of Production Research*, 38(13):3031–3041, 2000.
- [Jor98] C. Jordan. A two-phase genetic algorithm to solve variants of the batch sequencing problem. *International Journal of Production Research (UK)*, 36(3):745–760, 1998.
- [KKT87] A. Kolen, A. Rinnooy Kan, and H. Trienekens. Vehicle routing with time windows. *Operations Research*, 35:266–273, 1987.
- [KN01] Y. Karuno and H. Nagamochi. A 2-approximation algorithm for the multi-vehicle scheduling problem on a path with release and handling times. In *Proceedings of the European Symposium on Algorithms*, pages 218–229, 2001.
- [KR92] M. Kantor and M. Rosenwein. The orienteering problem with time windows. *Journal of the Operational Research Society*, 43:629–635, 1992.
- [Sav85] M. Savelsbergh. Local search for routing problems with time windows. *Annals of Operations Research*, 4:285–305, 1985.
- [Tha95] S. Thangiah. *Vehicle Routing with Time Windows using Genetic Algorithms, Application Handbook of Genetic Algorithms: New Frontiers, Volume II. Lance Chambers (Ed.)*. CRC Press, 1995.
- [TLZ00] K.C. Tan, L.H. Lee, and K.Q. Zhu. Heuristic methods for vehicle routing problem with time windows. In *Proceedings of the 6th AI and Math*, 2000.
- [TOVS93] S. R. Thangiah, I. H. Osman, R. Vinayagamoorthy, and T. Sun. Algorithms for the vehicle routing problems with time deadlines. *American Journal of Mathematical and Management Sciences*, 13(3&4):323–355, 1993.
- [Tsi92] J. Tsitsiklis. Special cases of traveling salesman and repairman problems with time windows. *Networks*, 22:263–282, 1992.
- [WS95] J. Wisner and S. Siferd. A survey of u.s. manufacturing practices in make-to-order machine shops. *Production and Inventory Management Journal*, 1:1–7, 1995.
- [YL99] W. Yang and C. Liao. Survey of scheduling research involving setup times. *International Journal of Systems Science*, 30(2):143–155, 1999.