Online and Offline Algorithms for the Time Dependent TSP with Time Zones

Björn Brodén[1], Mikael Hammar[2], and Bengt J. Nilsson[3]

[1] Department of Computer Science, Lund University,
Box 118, S-221 00 Lund, Sweden

[2] Dipartimento di Informatica ed Applicazioni, Università di Salerno
Baronissi (SA) - 84081, Italy

[3] School of Technology and Society, Malmö University College,
SE-205 06 Malmö, Sweden

Abstract. The time dependent traveling salesman problem is a variant of TSP with time dependent edge costs. We study some restrictions of TDTSP where the number of edge cost changes are limited. We find competitive ratios for online versions of TDTSP. From these we derive polynomial time approximation algorithms for graphs with edge costs one and two. In addition, we present an approximation algorithm for the orienteering problem with edge costs one and two.

1 Introduction

Transportation and scheduling problems modeled by the traveling salesman problem are inherently static. To model a dynamic environment a generalization is needed that incorporates changes in the environment. Such a generalization is provided by the time dependent traveling salesman problem (TDTSP). This problem is a generalization of TSP in which the cost of each edge depends on time, i.e., the cost of an edge depends on the time interval during which the edge is traversed. Several aspects of TDTSP have been studied and it seems to be difficult to collect them all under a single definition. The many variations all stem from the way that time is being modelled.

In some definitions time is proportional to the cost of the edges traversed [12]. This gives a natural generalization of TSP and is suitable if for example variations in the traffic load are important in computing a TSP tour for a delivery company. It also has connections to other generalizations of TSP, such as the kinetic TSP where moving points in the plane [9, 10] or on a line [10] are considered. We call this formulation *the cost dependent traveling salesman problem* (CDTSP).

Our focus is on another formulation of TDTSP where the cost of an edge depends on its position in the path. We call this problem the *step dependent traveling salesman problem*, denoted SDTSP. This interpretation has been studied on numerous occasions by the operations research community, due to its application in scheduling [3, 5, 6, 14]. We limit our study of TDTSP to instances where the edge costs are restricted in size and can change only a limited number of times.

Due to complications caused by time dependencies, approximation algorithms for TDTSP are hard to analyze. By considering online algorithms part of the time dependency is removed and a more tractable structure is achieved. Online algorithms for TSP have been studied in the past [2, 4, 11]. These algorithms take a set of cities as input, and compute the shortest TSP-tour. Over time, new cities are being added to the set. We are studying TSP in a dynamic environment; although the cities are known from the beginning, the distance between a pair of cities change over time.

The online algorithms we present here still require a solution to the NP-hard Orienteering problem [1]. As we shall see in Section 5 this problem allows a 3/4-approximation algorithm for graphs with edge costs one and two. We construct a polynomial time approximation algorithm for SDTSP by choosing the best answer given by a set of online algorithms. This results in a $2-2/3k$-approximation algorithm for the SDTSP where the edge costs are restricted to the values one and two and the costs can change at most $k-1$ times. The restrictions on the edge costs can be removed given an Orienteering algorithm that handles arbitrary edge costs. Since the inapproximability ratio of TSP grows with the relative edge costs we cannot expect algorithms for SDTSP with approximation ratio independent of the costs.

In Section 2 we state the formal definition of SDTSP. In Section 3 we analyze the online version of the problem and in Section 4 we consider polynomial time approximation algorithms. We also give an approximation algorithm for the Orienteering problem for edge costs one and two. In Section 5 we consider the cost dependent traveling salesman problem. We give an inapproximability result for the Euclidean CDTSP and present an online algorithm for CDTSP with two time zones and edge costs one and two. This online algorithm is modified to give an approximation algorithm as in the previous section.

2 Definition of SDTSP

Definition 1. *Consider a set of edge cost functions $\{c_1, \ldots, c_n\}$ assigned to a complete graph $G = (V, E)$ with $|V| = n$, $|E| = \binom{n}{2}$, and where $c_t(e)$ is the cost function for edge $e \in E$. Let e_{ij} denote the edge between v_i and v_j in V. The* step dependent traveling salesman problem (SDTSP) *seeks a permutation π of V that minimizes*

$$c_n(e_{\pi_n \pi_1}) + \sum_{i=1}^{n-1} c_i(e_{\pi_i \pi_{i+1}}).$$

In order to comprehend Definition 1 it helps to consider an instance of SDTSP as an n-layered graph, each layer containing n vertices. Layer i and layer $i+1$ form a complete bipartite graph where the edges are directed, going from layer i to layer $i+1$, and are given weights according to weight function c_i. The objective is to find a path going from layer one to layer n that visits all columns, starting from and ending at the same column.

We can view c_t as a discrete time dependent edge cost function defined on $t = \{1, \ldots, n\}$. To simplify the problem we restrict our study to the case where this cost function changes at most $k-1$ times as a function of t. A region where the cost function is constant is called a time zone. Let $0 = z_0, z_1, \ldots, z_{k-1}, z_k = n \in \mathbb{N}$ denote the time zone divisors, i.e., time zone $i = \{z_{i-1}+1, \ldots, z_i\}$, and $c_j = c_{j'}$ for $z_{i-1} < j, j' \leq z_i$. We simplify the representation by assigning one cost function to each time zone, giving us an instance $I = \{(c_1, z_1), \ldots, (c_k, z_k)\}$. An instance of SDTSP with k time zones is denoted SD^{k}TSP. Let M and m denote the costs of the most and least expensive edges using any of the cost functions $c_1, \ldots, c_k$.

We primarily study the online version of SD^{k}TSP for which an algorithm receives information regarding the time zones and the cost functions online. Let $I_j = (c_j, z_j)$ represent the instance restricted to time zone j, i.e., $I = \{I_1, \ldots, I_k\}$. The algorithm receives the input instance one time zone at a time. For each time zone j it produces a path P_j that contains $z_j - z_{j-1}$ previously unvisited vertices and with edge weights given by c_j. When P_j has been computed, the algorithm receives the next part of the input instance, i.e., I_{j+1}. After k time zones the algorithm has received the entire input instance and a Hamilton cycle in G has been built.

Definition 2. *Given an arbitrary algorithm A for* SD^{k}TSP *we write $A[I]$ for the vertices of the cycle that the algorithm chooses on the instance I and $A(I)$ for the cost of the resulting cycle produced on I. This will also be used for specific time zones, for instance $A(I_1)$ is the cost of the path in time zone one.*

Let $A^j[I] = \cup_{i=1}^{j} A[I_i]$, i.e., the vertices that A chooses from time zone one to time zone j.

Definition 3. *Let $R(A)$ denote the competitive ratio for algorithm A and R the smallest competitive ratio for any online algorithm, i.e., $R = \min_A\{R(A)\}$. We use* ALG *to denote an arbitrary online algorithm,* OFF *an arbitrary offline algorithm and* OPT *the optimal offline algorithm for* SD^{k}TSP.

3 Online SD^{k}TSP

Here we present a lower bound on the competitive ratio for SD^{k}TSP. We also present a strategy with a competitive ratio matching the lower bound. To compute the lower bound we use an adversary argument. The adversary builds a hard instance for the strategy. This is done online in response to the decisions made by the strategy. Let I^z denote the instance created by the adversary. In addition to this instance we define an adversary offline algorithm ZIG. This algorithm is adapted both to I^z and to the decisions that were made by the online strategy being analyzed. We can think of the adversary algorithm as being run in hindsight after the completion of the online strategy. Note that ZIG is designed to be optimal for I^z.

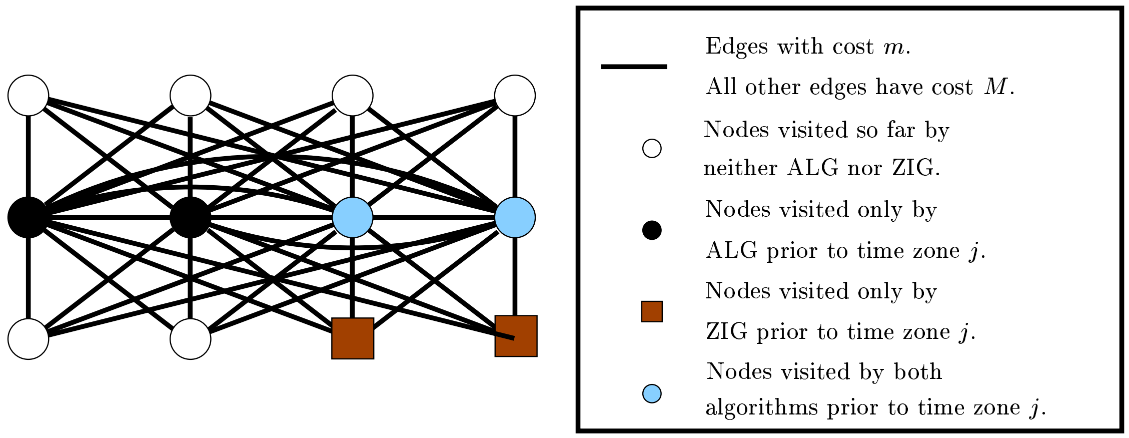
Edges with cost m.
All other edges have cost M.
Nodes visited so far by neither ALG nor ZIG.
Nodes visited only by ALG prior to time zone j.
Nodes visited only by ZIG prior to time zone j.
Nodes visited by both algorithms prior to time zone j.

Fig. 1. Time zone j of a typical instance created by the adversary.

The adversary constructs the instance I^z as follows: all edge costs in I_1^z are set to m. In I_j^z, for $2 \leq j \leq k$, all costs of edges adjacent to vertices in ALG$^{j-1}[I^z]$ are set to m and all other edge costs are set to M. Time zone j of a typical instance I^z is shown in Figure 1. The algorithm ZIG is designed to maximize the number of cost m edges used. It is described below and its strategy for time zone j is illustrated in Figure 2.

```
Algorithm ZIG
Input: The time zone information $z_0, \ldots, z_k$ and the Hamilton path $P_A$ produced by ALG.
Output: A Hamilton path $P_Z$.
    $P_Z = \emptyset$; $bottom := 1$; $top := n$
    for $i := 1$ to $z_1$ do
        $P_Z[i] := P_A[top]$; $top := top-1$
    endfor
    for $j := 2$ to $k$ do
        /*for each time zone $j > 1$*/
        for $i := z_{j-1}+1$ to $z_j$ do
            if $bottom < z_{j-1}$ and $odd(i)$ then
                $P_Z[i] := P_A[bottom]$; $bottom := bottom+1$
            else
                $P_Z[i] := P_A[top]$; $top := top-1$
            endif
        endfor
    endfor
End ZIG
```

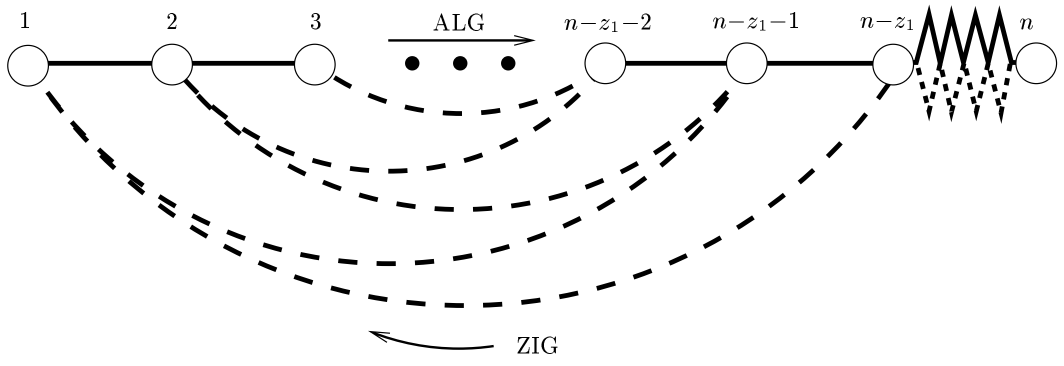

Fig. 2. Description of the strategy used by ZIG.

To determine the competitive ratio we need to measure the number of cost m edges that are used by ZIG. Especially, we would like to count the number of cost m edges that are used by ZIG but not by ALG. The usage function is a general concept that counts the number of cost m edges used by an arbitrary offline algorithm OFF up to a specified time zone j. A counted edge e should comply with the following conditions: (1) If the online algorithm uses e in time zone $l < j$ then $c_l(e) = M$. (2) The online algorithm cannot use e in a time zone $l \geq j$.

To make sure that the second condition holds we only count edges that contain at least one endpoint visited by ALG in a previous time zone. To keep track of these edges we define the concept of black vertices.

Definition 4. *The* black vertices *in zone j are defined as*

$$V_j^b(\mathrm{OFF}) = \mathrm{ALG}^{j-1}[I] \setminus \mathrm{OFF}^{j-1}[I].$$

We assume that an online strategy ALG has been run on an instance I resulting in a cycle P_A. We label the vertices with unique numbers from one up to n according to their position in the cycle P_A.

Definition 5. *The usage function $U_j(P)$ for a path P in a* SD^{k}TSP *instance, is defined as follows:*

$$U_1(P) = 0$$
$$U_j(P) = U_{j-1}(P) + |\{e \in P_j \mid c_j(e) = m \textbf{ and } e \textit{ has an endpoint in } V_j^b(\mathrm{OFF})\}|$$

Here P_j denotes the subpath of P restricted to time zone j. We will violate the notation and use $U_j(A)$ to denote the usage function for a cycle produced by the algorithm A. We let $U_j = U_j(P_Z)$, P_Z being the cycle produced by ZIG run on I^z, and we use V_j^b to denote the coloring of the vertices in time zone j, using ZIG as the offline algorithm on I^z. We want to find the value of U_j. Note that the usage function does not count edges chosen in time zone one.

Lemma 1. *If* ALG *visits a vertex in time zone $j > 1$ that has been visited by* ZIG *in a previous time zone then $|\mathrm{ALG}^j[I] \cup \mathrm{ZIG}^j[I]| = n$, where I is an arbitrary input instance.*

Proof. Assume that ALG visits the vertex labelled i during time zone j, and that ZIG visits vertex i during time zone j', where $j' < j$. Consider algorithm ZIG at the point when it visits vertex number i. By construction, *bottom* $\leq z_{j'}$. Hence, $P_A[\mathit{bottom}]$ has already been visited by ALG which implies that $i \neq$ *bottom*. Thus, $i =$ *top*. This means that all vertices $P_A[l]$, for $l \geq i$ have already been visited by ZIG. After time zone j all vertices $P_A[l]$, for $l \leq i$ have been visited by ALG, by the assumption above. We therefore conclude that $|\mathrm{ALG}^j[I] \cup \mathrm{ZIG}^j[I]| = n$. □

The number of cost m edges that ZIG use in time zone j is limited by the time zone's length and $2|V_j^b|$ as follows.

Lemma 2. *The number of cost m edges counted by $U_j(\mathrm{OFF})$ in time zone $j > 1$ is at most*

$$\min\{2|V_j^b(\mathrm{OFF})|, z_j - z_{j-1}\}.$$

Proof. Let P be the cycle generated by OFF and P_j the subpath of P in time zone j. All edges counted by $U_j(\mathrm{OFF})$ in time zone j belong to P_j by definition, and $|P_j| = z_j - z_{j-1}$. Secondly, all edges counted by $U_j(\mathrm{OFF})$ in time zone j have at least one endpoint in $V_j^b(\mathrm{OFF})$. This implies that each vertex in $V_j^b(\mathrm{OFF})$ is adjacent to at most 2 counted edges. It follows that the number of cost m edges counted is at most $2|V_j^b(\mathrm{OFF})|$. □

Lemma 3. *The number of cost m edges counted by U_j in time zone $j > 1$ is exactly*

$$\min\{2|V_j^b|, z_j - z_{j-1}\}.$$

Proof. From Lemma 2 it follows that ZIG uses at most $\min\{2|V_j^b|, z_j - z_{j-1}\}$ cost m edges in time zone j. To see that ZIG uses at least $\min\{2|V_j^b|, z_j - z_{j-1}\}$ cost m edges of I^z we need to examine the algorithm in detail.

If $|\mathrm{ALG}^{j-1}[I^z] \cup \mathrm{ZIG}^j[I^z]| = n$ then all edges taken by ZIG in time zone j have endpoints in $\mathrm{ALG}^{j-1}[I^z]$ and by definition, these edges have cost m in I^z. Hence, exactly $z_j - z_{j-1}$ cost m edges are counted by U_j in time zone j.

If $|\mathrm{ALG}^{j-1}[I^z] \cup \mathrm{ZIG}^j[I^z]| < n$ then there are vertices neither visited by ALG in a time zone prior to j nor by ZIG in a time zone prior to $j+1$. Let t_j denote the value of *top* and b_j the value of *bottom* at the beginning of time zone j and let t'_j and b'_j denote the corresponding values at the end of the time zone. Now, z_{j-1} is the index of the last vertex visted by ALG in time zone $j-1$, and since there are unvisited vertices left, $t'_j > z_{j-1}$. The black vertices therefore have the labels $b_j, \ldots, z_{j-1}$. Thus, $z_{j-1} - b_j + 1 = |V_j^b|$. Furthermore, any vertex taken from the top of P_A has not been visited yet by ALG.

Consider algorithm ZIG at the beginning of time zone j. Every time *bottom* is increased, a black vertex is incorporated into ZIG's path. Every time *top* is decreased a vertex unvisited by ALG is inserted into the path. Thus, every second vertex picked is black save perhaps a string of uncolored vertices at the end. If exactly every second vertex in the path constructed for time zone j is black then all $z_j - z_{j-1}$ edges are counted by U_j. If less than every second vertex is black then the if-statement inside the for-loop has been violated more than $(z_j - z_{j-1})/2$ times and we infer that $b'_j = z_{j-1}$. Thus, *bottom* has been increased $|V_j^b|$ times and all black vertices lie on the subpath of P_Z in time zone j. For every black vertex two edges are counted by U_j. In this case $2|V_j^b|$ edges are counted. □

Lemma 2 gives the following upper bound on $U_j(\mathrm{OFF})$.

Lemma 4. $U_j(\mathrm{OFF})$ *is given by the following recurrence relation:*

$$\begin{aligned} &U_1(\mathrm{OFF}) \le 0 \\ &U_j(\mathrm{OFF}) \le U_{j-1}(\mathrm{OFF}) + \min\left\{2|V_j^b(\mathrm{OFF})|, z_j - z_{j-1}\right\}, \textit{ if } 2 \le j \le k, \end{aligned}$$

Lemma 3 gives us a recurrence relation for U_j similar to the upper bound on $U_j(\mathrm{OFF})$.

Lemma 5. U_j *is given by the following recurrence relation:*

$$\begin{aligned} &U_1 = 0 \\ &U_j = U_{j-1} + \min\left\{2|V_j^b|, z_j - z_{j-1}\right\}, \textit{ if } 2 \le j \le k, \end{aligned}$$

To evaluate the recurrence we would like to express $|V_j^b|$ in terms of U_{j-1} and z_{j-1}. To this end we use the following lemma.

Lemma 6. *If* $2|V_j^b| < z_j - z_{j-1}$ *then* $\left|\mathrm{ALG}^{j-1}[I^z] \cup \mathrm{ZIG}^{j-1}[I^z]\right| < n$.

Proof. At the beginning of time zone j, the number of vertices visited by ALG but not by ZIG is $|V_j^b|$. The number of vertices visited by ZIG in previous time zones is z_{j-1}. Therefore,

$$\begin{aligned} \left|\mathrm{ALG}^{j-1}[I^z] \cup \mathrm{ZIG}^{j-1}[I^z]\right| &= \left|\mathrm{ALG}^{j-1}[I^z] \setminus \mathrm{ZIG}^{j-1}[I^z]\right| + \left|\mathrm{ZIG}^{j-1}[I^z]\right| \\ &= |V_j^b| + z_{j-1} \\ &\le 2|V_j^b| + z_{j-1} \\ &< z_j - z_{j-1} + z_{j-1} = z_j \le n. \end{aligned}$$

□

Lemma 7. *If* $2|V_j^b| < z_j - z_{j-1}$ *then* $U_{j-1} = 2\left|\mathrm{ALG}^{j-1}[I^z] \cap \mathrm{ZIG}^{j-1}[I^z]\right|$.

Proof. If $2|V_j^b| < z_j - z_{j-1}$ then $\left|\mathrm{ALG}^{j-1}[I^z] \cup \mathrm{ZIG}^{j-1}[I^z]\right| < n$, according to Lemma 6. From Lemma 1 it follows that prior to time zone j, ALG has never visited a vertex already visited by ZIG. We infer two consequences from this fact, the first being that all vertices in $\mathrm{ALG}^{j-1}[I^z] \cap \mathrm{ZIG}^{j-1}[I^z]$ were black when ZIG visited them. The second consequence is that $top > z_{j-1}$ at the end of time zone $j-1$. This implies that no edge in the path produced by ZIG has two endpoints in $\mathrm{ALG}^{j-1}[I^z] \cap \mathrm{ZIG}^{j-1}[I^z]$. The number of edges incident to vertices in $\mathrm{ALG}^{j-1}[I^z] \cap \mathrm{ZIG}^{j-1}[I^z]$ is therefore $2\left|\mathrm{ALG}^{j-1}[I^z] \cap \mathrm{ZIG}^{j-1}[I^z]\right|$ and all of them are counted by U_{j-1}, since U_{j-1} only counts edges that ALG has already visited. Hence, the result follows. □

Now we can find a simpler expression for U using the following transformation. If $2|V_j^b| < z_j - z_{j-1}$ then $|V_j^b| = \left|\mathrm{ALG}^{j-1}[I^z]\right| - \left|\mathrm{ALG}^{j-1}[I^z] \cap \mathrm{ZIG}^{j-1}[I^z]\right| = z_{j-1} - U_{j-1}/2$, by Lemma 7. We have proved the following theorem.

Theorem 1. *The usage function of* ZIG *is equal to*

$$\begin{aligned} &U_1 = 0, \\ &U_j = U_{j-1} + \min\left\{2z_{j-1} - U_{j-1}, z_j - z_{j-1}\right\}, \text{ if } 2 \le j \le k. \end{aligned}$$

Given Theorem 1 we compute a maximum value of U_k.

Theorem 2. *The maximum value of the function* U_k *is* $U_k = n - z_1$ *and occurs when* $z_i = (2^i - 1)n/(2^k - 1)$.

Proof. Assuming $U_0 = 0$ we have that

$$U_j = U_{j-1} + \min\left\{2z_{j-1} - U_{j-1}, z_j - z_{j-1}\right\}.$$

U_j is maximized if $2z_{j-1} - U_{j-1} = z_j - z_{j-1}$. That is,

$$U_j = 2z_{j-1}, \tag{3.0.1}$$

$$U_j = U_{j-1} + z_j - z_{j-1}. \tag{3.0.2}$$

Substituting U_j with $2z_{j-1}$ in (3.0.2) yields

$$z_j - 3z_{j-1} + 2z_{j-2} = 0.$$

We solve the equation given that $z_0 = 0$ and $z_k = n$:

$$z_j = \frac{2^j - 1}{2^k - 1} n.$$

After substituting z_j accordingly in (3.0.1) we have that

$$U_k = n - z_1,$$

since $z_1 = n/(2^k - 1)$. □

We state a final lemma concerning the usage function for any offline algorithm OFF. We use this to prove upper bounds on online algorithms for SD^{k}TSP.

Lemma 8. *Let* $0{=}z_0, z_1, \ldots, z_j {\le} n \in \mathbb{N}$. *For any instance* $I = \{(c_1, z_1), \ldots, (c_j, z_j)\}$ *of* $\mathrm{SD}^k\mathrm{TSP}$ *and any offline algorithm* OFF,

$$U_j(\mathrm{OFF}) \le U_j$$

Proof. By definition $U_1(\mathrm{OFF}) = U_1 = 0$.

For $j > 1$, assume that $U_{j-1}(\mathrm{OFF}) \le U_{j-1}$. By definition, U_{j-1} counts the number of low cost edges visited by OFF in time zone $j' < j$ having at least one end point in $\mathrm{ALG}^{j'-1}[I]$. The number of such vertices is at least $\frac{U_{j-1}}{2}$, and hence,

$$\begin{aligned}\frac{U_{j-1}}{2} &\le \left|\mathrm{ALG}^{j'-1}[I] \cap \mathrm{OFF}^{j'}[I]\right| \\ &\le \left|\mathrm{ALG}^{j-1}[I] \cap \mathrm{OFF}^{j-1}[I]\right| \\ &= \left|\,\left|\mathrm{ALG}^{j-1}[I]\right| - \left|\mathrm{ALG}^{j-1}[I] \setminus \mathrm{OFF}^{j-1}[I]\right|\right. \\ &= z_{j-1} - \left|V_j^b(\mathrm{OFF})\right|.\end{aligned}$$

From Lemma 4 and our induction hypothesis we have that

$$\begin{aligned}U_j(\mathrm{OFF}) &\le U_{j-1}(\mathrm{OFF}) + \min\left\{2|V_j^b(\mathrm{OFF})|, z_j - z_{j-1}\right\} \\ &\le U_{j-1}(\mathrm{OFF}) + \min\left\{2z_{j-1} - U_{j-1}(\mathrm{OFF}), z_j - z_{j-1}\right\} \\ &= \min\left\{2z_{j-1}, z_j - z_{j-1} + U_{j-1}(\mathrm{OFF})\right\} \\ &\le \min\left\{2z_{j-1}, z_j - z_{j-1} + U_{j-1}\right\} \\ &= U_{j-1} + \min\left\{2z_{j-1} - U_{j-1}, z_j - z_{j-1}\right\} \\ &= U_j\end{aligned}$$

□

3.1 Lower Bound on the Competitive Ratio

This section uses the results we have arrived at so far to state the first of our main results.

Theorem 3. *The competitive ratio, R, of any online algorithm for* $\mathrm{SD}^k\mathrm{TSP}$ *is*

$$R \ge 1 + \frac{(M-m)U_k}{\mathrm{ZIG}(I^z)}$$

Proof. Consider an arbitrary online strategy ALG. To analyze the competitive ratio we use our adversary argument, including the instance I^z together with the adversary algorithm ZIG. The online algorithm has to pay M on all edges except those used in time zone one, since edges with cost set to m in other zones can only be used by ZIG. The cost for ZIG on the instance I^z is $\mathrm{ZIG}(I^z) = mU_k + M(n - (U_k + z_1)) = Mn - (M-m)(U_k + z_1)$, giving us the competitive ratio

$$R = \min_{\mathrm{ALG}} \max_I \frac{\mathrm{ALG}(I)}{\mathrm{OPT}(I)} \ge \min_{\mathrm{ALG}} \frac{\mathrm{ALG}(I^z)}{\mathrm{ZIG}(I^z)} \ge \frac{M(n-z_1) + mz_1}{\mathrm{ZIG}(I^z)}.$$

Doing the calculations backwards we get that

$$\begin{aligned}R &\ge \frac{M(n-z_1) + mz_1 - Mn + (M-m)(U_k + z_1) + \mathrm{ZIG}(I^z)}{\mathrm{ZIG}(I^z)} \\ &= 1 + \frac{(M-m)U_k}{\mathrm{ZIG}(I^z)}\end{aligned}$$

□

Applying Theorem 2 to Theorem 3 produces the following corollary.

Corollary 1. *The worst case competitive ratio R of any strategy* ALG *on any instance* $I = \{(c_1, z_1), \ldots, (c_k, z_k)\}$ *is at least*

$$R \geq \frac{M}{m} \cdot \frac{2^k - 2}{2^k - 1} + \frac{1}{2^k - 1}.$$

3.2 Upper Bound on the Competitive Ratio

There is a simple strategy that achieves the upper bound found in the last section. This algorithm, presented below, uses a greedy approach.

```
Algorithm   GREEDY for SD$^k$TSP
Input:   A sequence of $k$ instances $I_1, ..., I_k$.
Output:  A Hamilton cycle $P$.
1  Given an instance $I_j$, produce the cheapest path of size $z_j - z_{j-1}$ on the unvisited vertices in $V$.
End  GREEDY for SD$^k$TSP
```

Theorem 4. *The competitive ratio of* GREEDY *as a function of U_k is at most*

$$1 + \max_I \frac{(M - m)U_k}{\text{OPT}(I)}$$

Proof. Let I be an arbitrary SD^{k}TSP instance, and P_I the optimal TSP tour on I. In time zone one GREEDY(I_1) = OPT(I_1). Assume that the following holds for $j > 1$:

$$\text{GREEDY}(I_j) \leq \text{OPT}(I_j) + (M - m)(U_j(P_I) - U_{j-1}(P_I)).$$

Summing up the inequalities (including time zone one) yields that

$$\begin{aligned}\sum_{i=1}^{k} \text{GREEDY}(I_i) &\leq \text{OPT}(I) + (M - m)(U_k(P_I)) \\ &\leq \text{OPT}(I) + (M - m)U_k,\end{aligned}$$

according to Lemma 8. This gives a competitive ratio of

$$R(\text{GREEDY}) \leq \max_I \frac{\text{OPT}(I) + (M - m)U_k}{\text{OPT}(I)} = 1 + \max_I \frac{(M - m)U_k}{\text{OPT}(I)}.$$

It remains to prove our assumed inequality. This is equivalent to showing that there is a path usable by GREEDY in time zone $j > 1$ that costs at most OPT$(I_j) + (M - m)(U(\text{OPT}^j[I]) - U(\text{OPT}^{j-1}[I]))$. First observe that the number of edges counted by the usage function in time zone j is $U_j(P_I) - U_{j-1}(P_I)$. OPT pays at least $m(U_j(P_I) - U_{j-1}(P_I))$ for these edges. This cost is included in OPT$^j[I]$. Edges not counted by the usage function can be used by ALG for the same cost as OPT. These edges are connected with possibly expensive edges for a total cost of OPT$(I_j) + (M - m)(U_j(P_I) - U_{j-1}(P_I))$; see Figure 3. By construction GREEDY finds a path having at most this cost. □

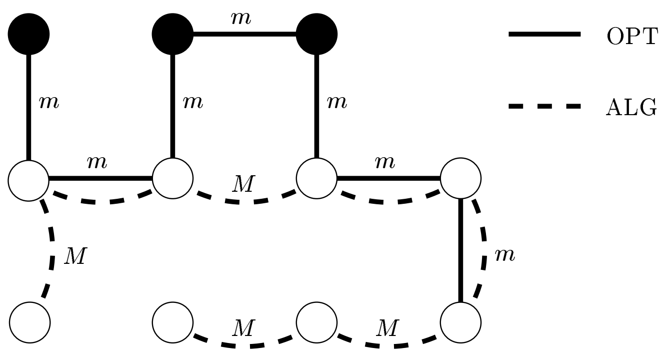

Fig. 3. Patching together the pieces of the optimal algorithm's path.

This theorem together with Theorem 3 gives a sharp worst case competitive ratio.

Corollary 2. *The competitive ratio of* GREEDY *is*

$$R = \frac{M}{m} \cdot \frac{2^k - 2}{2^k - 1} + \frac{1}{2^k - 1}.$$

Note that we have no implementation of GREEDY that runs in polynomial time, since it contains an NP-complete subproblem.

4 Polynomial Time Approximation Algorithms for SD^{k}TSP

Next we describe a polynomial time algorithm for SD^{k}TSP with edge costs from the set $\{1, 2\}$. We use the exponential time greedy algorithm designed for the online case. This greedy algorithm is exponential since it finds an optimal k-TSP path. We give an approximation algorithm for the longest path problem in a graph with edge costs in the set $\{1, 2\}$. This problem is also referred to as the Orienteering problem in the literature [1]. The new algorithm gives an approximation to k-TSP, which can be used to make the greedy algorithm polynomial.

4.1 Orienteering with Edge Costs One and Two

Definition 6. *Given a value T and a graph G with edge weights one and two, the* Orienteering(1, 2) problem *is to compute the longest path in G with cost at most T. The length of a path is the number of edges it consists of.*

Let us start with a straightforward algorithm that achieves an approximation factor of 2/3. Then we describe a simple enhancement, improving the approximation factor to 3/4.

The algorithm is based on matching. We simply perform a maximum matching on the cost one edges in the input graph G. Observe that the matching consists of at least half the number of cost one edges in the optimal path. Hence, we construct a path with cost T in which every second edge comes from the matching as long as there are edges in the matching left to choose. Thereby we guarantee that the length of our path is at least two thirds of the optimal length.

This simple algorithm can be improved if we perform a second maximum matching using the cost one edges that are left after the first matching. We observe that the matched edges from both matchings induce a possibly disconnected subgraph of G containing paths of lengths between one and n and cycles of lengths between four and n. If we break up all the cycles into paths by removing one edge we transform the induced graph into a forest of paths. A cycle of length i is thereby transformed into a path of length $i-1$. Denote the resulting forest G'. Our algorithm builds a long path with cost T by concatenating the paths of G' in decreasing length order, using arbitrary edges as "glue".

```
Algorithm   Enhanced Orienteering$(1,2)$
Input:   A value $T$ and a complete graph $G=(V,E)$ with edge costs either one or two.
Output:  A path with cost $T$.
1  $E_1 \leftarrow$ A maximum matching on the cost one edges in $G$.
2  $E_2 \leftarrow$ A maximum matching on the remaining cost one edges in $G$.
3  $G' \leftarrow (V, E_1 \cup E_2)$
4  Break up the cycles in $G'$ as described above.
5  Construct a Hamilton path $H$ of $G$ by concatenating the paths in $G'$ in decreasing length order,
   appending the remaining cost two edges to the end.
6  return  the longest (initial) part of $H$ with cost less or equal to $T$.
End  Enhanced Orienteering$(1,2)$
```

The time complexity of this algorithm is dominated by the maximum matching procedure, which is polynomial [7].

To confirm the approximation ratio we compare the path built by the enhanced algorithm (APX) with the path constructed by an optimal algorithm (OPT). Let p_i denote the number of paths with length i in G'. Let the length of the shortest path from G' used in APX be j and let x_j denote the number of length j paths used in APX. Also, let p_0 denote the number of consecutive cost two edges in APX. The optimal path contains a corresponding set of paths built up by consecutive cost one edges. We divide these paths into a set K_2 containing pairs of consecutive edges and a set K_1 containing the remaining loose edges as described in Figure 4. Let k_2 denote the number of edge pairs in K_2, k_1 the number of edges in K_1, and k_0 the number of cost two edges in OPT. The lengths of APX and OPT are

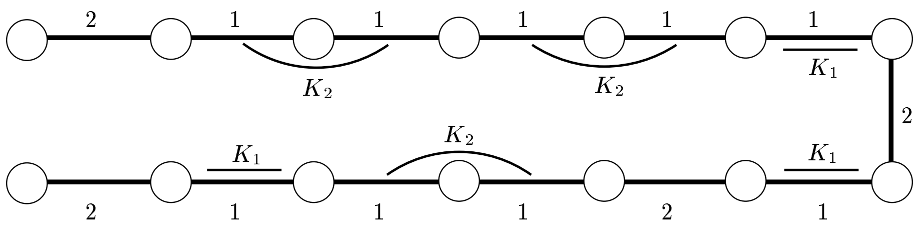

Fig. 4. Divide the paths of cost one edges within OPT into the sets K_1 and K_2 as described above.

$$\mathrm{OPT} = 2k_2 + k_1 + k_0, \tag{4.1.1}$$
$$\mathrm{APX} = \sum_{i>j}(i+1)p_i + (j+1)x_j + p_0, \tag{4.1.2}$$

and the value T can be expressed as

$$T = \mathrm{OPT} + k_0, \tag{4.1.3}$$
$$T \leq \sum_{i>j}(i+2)p_i + (j+2)x_j + 2p_0. \tag{4.1.4}$$

Let A denote the number of cost one edges in G' that originate from the first maximum matching. Let B denote the number of cost one edges from the second matching in G'. The following Lemma describes crucial relations regarding the number of edges matched by the algorithm, the number of paths in G' and the number of cost one edges in OPT.

Lemma 9. *The following four conditions hold.*

$$B \leq \sum_{i>1}(i-1)p_i, \tag{4.1.5}$$
$$A > \frac{3k_1 + 6k_2}{8}, \tag{4.1.6}$$
$$B \geq k_2, \tag{4.1.7}$$
$$A + B = \sum_{i\geq 1} ip_i. \tag{4.1.8}$$

Proof. (4.1.5) $B = \sum_{i>1}\lfloor i/2 \rfloor p_i$, since every second edge in a path from G' comes from the second matching. Furthermore, since $\lfloor i/2 \rfloor \leq i-1$ if $i \geq 1$,

$$\sum_{i>1}\lfloor i/2 \rfloor p_i \leq \sum_{i>1}(i-1)p_i.$$

(4.1.6) The first matching will include at least half the number of cost one edges in the optimal path, i.e. $(2k_2+k_1)/2$. But as we stated in the algorithm, some of the edges from the first matching can be lost as we break up the cycles. But each cycle has at least four edges, which means that we can loose at most a quarter of the edges from the matching. Thus,

$$A \geq \frac{3}{4}\cdot\frac{2k_2+k_1}{2} = \frac{6k_2+3k_1}{8}$$

(4.1.7) The first matching will include at most one of the edges in each pair in K_2. The rest of the k_2 edges in K_2 are left for the second matching, and all of them can be used for the matching. Thus, the number of edges used in the second matching is at least k_2.

The correctness of (4.1.8) follows directly from the definition of p_i, A and B. □

To prove the approximation ratio we need to consider two cases: $p_0 = 0$ and $p_0 \neq 0$.

Case 1: $p_0 = 0$.
From (4.1.3) and (4.1.4) it follows that

$$x_j \geq \frac{\mathrm{OPT} + k_0 - \sum_{i>j}(i+2)p_i}{j+2}.$$

Thus,

$$\begin{aligned}\text{APX} &\geq \sum_{i>j}(i+1)p_i + (j+1)\frac{\text{OPT} + k_0 - \sum_{i>j}(i+2)p_i}{j+2} \\ &= \frac{\sum_{i>j} p_i((j+2)(i+1)-(j+1)(i+2)) + (j+1)(k_0 + \text{OPT})}{j+2}.\end{aligned}$$

If $j = 1$ then

$$\sum_{i>j} p_i((j+2)(i+1)-(j+1)(i+2)) = \sum_{i>1}(i-1)p_i \geq B \geq k_2.$$

Inserting this into the expression for APX and taking the ratio between APX and OPT yields that

$$\frac{\text{APX}}{\text{OPT}} \geq \frac{k_2 + 2(k_0 + \text{OPT})}{3\text{OPT}}.$$

Since $k_0 \geq k_1$, and $\text{OPT} = 2k_2 + k_1 + k_0$ it follows that

$$\frac{\text{APX}}{\text{OPT}} \geq \frac{4k_2 + 4k_1 + 4k_0 + 8\text{OPT}}{12\text{OPT}} \geq \frac{10\text{OPT}}{12\text{OPT}} = 5/6.$$

If on the other hand $j \geq 2$ then $(j+2)(i+1)-(j+1)(i+2) \geq 0$, since $j \geq i > 0$. Therefore

$$\text{APX} \geq \frac{(j+1)\text{OPT}}{j+2}.$$

The approximation ratio is thus

$$\frac{\text{APX}}{\text{OPT}} \geq \frac{j+1}{j+2} \geq 3/4,$$

since, once again, $j \geq 2$.

Case 2: $p_0 \neq 0$.
Observe that $p_0 \neq 0$ implies that all edges in the matching are used in APX, i.e $j = 1$ and $x_1 = p_1$. Again we use (4.1.3) and (4.1.4) to get that

$$p_0 \geq \frac{\text{OPT} + k_0 - \sum_{i\geq 1}(i+2)p_i}{2}.$$

Inserting the expression for p_0 into (4.1.2) yields

$$\begin{aligned}\text{APX} &\geq \sum_{i\geq 1}(i+1)p_i + \frac{\text{OPT} + k_0 - \sum_{i\geq 1}(i+2)p_i}{2} && (4.1.9)\\ &= \frac{\sum_{i\geq 1} ip_i + \text{OPT} + k_0}{2} && (4.1.10)\\ &= \frac{A + B + \text{OPT} + k_0}{2} && (4.1.11)\\ &\geq \frac{\frac{6k_2+3k_1}{8} + k_2 + \text{OPT} + k_0}{2} && (4.1.12)\\ &= \frac{14k_2 + 3k_1 + 8k_0 + 8\text{OPT}}{16} && (4.1.13)\\ &\geq \frac{14k_2 + 5k_1 + 6k_0 + 8\text{OPT}}{1}6 && (4.1.14)\\ &= \frac{4k_2 + k_0 + 13\text{OPT}}{1}6 && (4.1.15)\\ &\geq 13/16 \cdot \text{OPT} && (4.1.16)\end{aligned}$$

In (4.1.11) and (4.1.12) we use Lemma 9, and in (4.1.14) we use once again the fact that $k_0 \geq k_1$. With this, Case 2 has been proved to hold, and Theorem 5 follows.

Theorem 5. *The enhanced algorithm has the approximation ratio* $3/4$.

From the analysis we infer that the worst case appears when $p_0 = 0$ and $j = 2$. The following example shows that the analysis is tight. The example contains a cost one skeleton of a graph, i.e., all cost one edges in the graph are shown in the figure. It is left for the reader to apply the enhanced algorithm on this example.

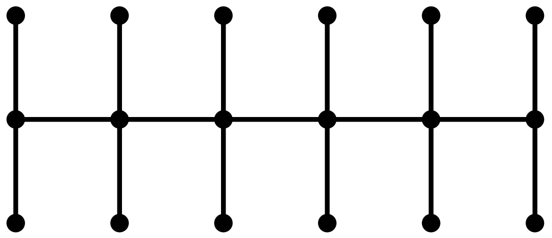

Fig. 5. Cost one skeleton of a worst case instance for the enhanced orienteering algorithm.

4.2 Polynomial Time Greedy Algorithms for SD^{k}TSP

Let us now define the polynomial time greedy algorithm (PG for short).

```
Algorithm    PG for SD^kTSP(1,2)
Input:    A sequence of $k$ instances $I_1, ..., I_k$.
Output: A Hamilton cycle $P$.
1  Given an element $I_j$, use a polynomial time algorithm for $(z_j - z_{j-1})$-TSP(1,2) on the unvisited
   vertices in $V$.
End  PG for SD^kTSP(1,2)
```

To simplify the analysis we define the concept of cost one ratio.

Definition 7. *Let* $\#_1(\text{ALG}[I])$ *be the number of cost one edges used by algorithm* ALG *on instance* I *and* $\#_1(\text{OPT}[I])$ *the number of cost one edges used by the optimal algorithm on the same instance. The cost one ratio of* ALG *is*

$$q_1(\text{ALG}) = \min_I \frac{\#_1(\text{ALG}[I])}{\#_1(\text{OPT}[I])}.$$

With q_1 we denote the largest known cost one ratio for any polynomial time SD^{k}TSP algorithm.

Theorem 6. *PG has a competitive ratio at most*

$$\frac{q_1 U_k + (2 - q_1)n}{n}$$

Proof. Let P_I denote the cycle produced by the new greedy algorithm. In every time zone, the number of cost one edges used by PG is at least q_1 times the number of cost one edges used by OPT that were not counted by the usage function.

This adds up to

$$\begin{aligned}&\frac{2U_k(P_I) + q_1(n - U_k(P_I)) + 2(1-q_1)(n - U_k(P_I))}{n}\\ =\ &\frac{q_1 U_k(P_I) + (2-q_1)n}{n} \leq \frac{q_1 U_k + (2-q_1)n}{n}.\end{aligned}$$

□

We can adopt the enhanced orienteering algorithm in the previous section to get a k-TSP path algorithm with $q_1 = \frac{2}{3}$ by returning a path of length T instead of a path with cost T. (To see that the algorithm attains this cost one ratio it is sufficient to consider its worst case, in which every third edge in the path has cost two when all edges in the optimal path have cost one.) Using PG with this new algorithm as a subroutine we get the competitive ratio

$$\frac{\frac{2}{3}U_k + \frac{4}{3}n}{n}.$$

Using the results in Theorem 2 we get the following largest competitive ratio:

$$\frac{2(2^k - 2)}{3(2^k - 1)} + \frac{4}{3}.$$

We present the competitive ratios for some values of k in Table 1.

Not surprisingly these values are rather close to 2. Note that the only cost one edges we can hope for in the worst case lie in time zone one. This situation does not change with a better orienteering algorithm. Thus, to improve the approximation ratio we need to modify the polynomial time greedy algorithm. We call the new algorithm IPG (improved greedy). To measure the performance of this

```
Algorithm   IPG for SD$^k$TSP(1,2)
Input:   A sequence of $k$ instances $I_1, ..., I_k$.
Output:  A Hamilton cycle $P$.
1  Use PG on every permutation of the set $\{I_1, ..., I_k\}$, and return the cheapest solution.
End  IPG for SD$^k$TSP(1,2)
```

algorithm we note that no offline algorithm can simultaneously force a value on the corresponding usage function U'_k higher than

$$U'_k = \min_{\pi}\{U(A_\pi)\},$$

where π is a permutation of $\{1, \ldots, k\}$, and A_π is the polynomial time greedy algorithm using the input sequence $I_{\pi_1}, I_{\pi_2}, \ldots, I_{\pi_k}$. A trivial upper bound on U'_k is if $z_j = \frac{jn}{k}$, for which $U'_k \leq \max_{z_i} = \frac{k-1}{k}n$. Using the same analysis as for PG we get that IPG has an approximation ratio no higher than

$$\frac{q_1 U'_k + (2-q_1)n}{n} \leq q_1\frac{k-1}{k} + 2 - q_1 = 2 - \frac{q_1}{k}.$$

With our best k-TSP(1,2) algorithm this simplifies to $2 - \frac{2}{3k}$. We print these results for small k in Table 1.

In contrast to the values of the GREEDY algorithm, that are sharp, the values for the IPG are only upper bounds. It is possible to make a small improvement on these results if we in the last time zone use a dedicated TSP(1,2) algorithm [13] instead of the the more general Orienteering algorithm.

k	GREEDY(PG with $q_1 = 1$)	PG with $q_1 = 2/3$	IPG with $q_1 = 1$	IPG with $q_1 = 2/3$
1	1.00	1.33	1.00	1.33
2	1.67	1.78	1.50	1.67
3	1.86	1.91	1.67	1.78
4	1.93	1.96	1.75	1.83
5	1.97	1.98	1.80	1.87
6	1.98	1.99	1.83	1.89

Table 1. Competitive ratio for SDTSP algorithms.

5 The Cost Dependent Traveling Salesman Problem

Let us end our exposition with the second variant of TDTSP: the cost dependent traveling salesman problem (CDTSP). In CDTSP the objective is to minimize the traveling time, which is identical to the total cost of the edges traversed.

Definition 8. *Consider a set of edge cost functions $\{c_1, \dots, c_{t_n}\}$ assigned to a complete graph $G = (V, E)$ with $|V| = n$, $|E| = \binom{n}{2}$, and where $c_t(e)$ is the cost function for edge $e \in E$. Let e_{ij} denote the edge between v_i and v_j in V. The* cost dependent traveling salesman problem (CDTSP) *seeks a permutation π of V that minimizes*

$$c_{t_1}(e_{\pi_1,\pi_2}) + c_{t_n}(e_{\pi_n,\pi_1}) + \sum_{i=2}^{n-1} c_{t_i}(e_{\pi_i,\pi_{i+1}}), \text{ where } t_i = \sum_{j=1}^{i-1} c_{t_j}(e_{\pi_j,\pi_{j+1}}).$$

The difference between SDTSP and CDTSP is that the cost function now depends on the weight sum of the edges already traversed by the salesman. Also in this case one can visualize the instance as a directed layered graph. However, an edge of cost c going from a vertex in layer i will connect with a vertex in layer $i + c$. Note, though, that the number of layers might be very large, e.g., equal to the length of the longest salesman path.

As for SDTSP we restrict the number of different time zones to k and denote the resulting problem CD^{k}TSP. We begin with a discouraging result for the Euclidean case.

Let $S = \{s_0, s_1, \dots, s_n\}$ be a set of points in the plane, with s_0 being the starting point and $z_0, z_1, \dots, z_{k-1}$ being the time zone divisors. Furthermore, let $C_1, \dots, C_k$ be k positive values such that $C_1 < \dots < C_k$. We define the function

$$C(t) = C_i, \text{ if } \tau_{i-1} \le t < \tau_i, \text{ for } 1 \le i \le k.$$

Consider a salesman that visits the points in S. We define the departure time t_i of point s_i as the point in time when the salesman leaves point s_i.

The *Restricted Euclidean CDTSP* is the problem of finding a tour that visits all the points in S, minimizing the total traveling time. The time needed to go between two points s_i and s_j is given by $d(s_i, s_j)C(t_i)$, where $d(s_i, s_j)$ is the distance between the two points. We assume that the departure time of the starting point is $t_0 = z_0 = 0$.

Garey, Graham and Johnson [8] prove that the Euclidean traveling salesman problem is NP-hard by a reduction from the NP-complete decision problem *exact cover by 3-sets*, X3C. We use their reduction to prove that the Restricted Euclidean CDTSP is inapproximable. If $\mathcal{F}$ is a yes instance of $X3C$, then we say that $\mathcal{F} \in X3C$, otherwise we say that $\mathcal{F} \notin X3C$.

Let S be an instance of TSP produced by Garey et al.'s reduction, such that $|S| = n$. The points of S lie on a unit grid G of size less that $n \times n$ and a naive tour visiting all points of S has a cost $l < 2n$. These results follow directly from Garey et al.'s construction.

Garey et al. prove that the cost of an optimal tour is less or equal to some specific value L^* if an exact cover exists for the X3C-instance. If there is no exact cover, then it has at least the cost $L^*{+}1$.

Theorem 7. *The Euclidean* $\mathrm{CD^2TSP}$ *cannot be approximated by any constant factor.*

Proof. We construct a Restricted Euclidean CDTSP-instance based upon the instance used in the reduction of Garey et al. Let r be an arbitrary constant. We take S from the reduction of Garey et al. as the set of points in the Restricted Euclidean CDTSP-instance, with the lower leftmost point as depot. We let $z_0 = 0$, $z_1 = L^*$, $C_1 = 1$, and $C_2 = (r{-}1)L^*$.

If $\mathcal{F} \in X3C$, then the time needed by an optimal salesman to visit all points and go back to the depot is L^* as in the TSP-instance of Garey et al. On the other hand, if $\mathcal{F} \notin X3C$ then the cost of the optimal tour in the TSP-instance is at least $L^*{+}1$. The shortest Hamilton path starting from the lower leftmost point is therefore at least L^* long. For the Restricted Euclidean CDTSP-instance this implies that the departure time of the last point s_i to be visited by the salesman is at least L^*. The cost of the last edge is therefore $(r{-}1)L^* d(s_i, s_0) \geq (r{-}1)L^*$, since $d(s_i, s_0) \geq 1$. The total traveling time is thus at least rL^* and the approximation ratio becomes at least $\frac{rL^*}{L^*} = r$. Since we can choose r arbitrarily large, the theorem follows. $\square$

Let us again restrict the edge costs to either one or two and consider the online version of $\mathrm{CD^2TSP}$. Once again the greedy algorithm is optimal.

Theorem 8. *The competitive ratio of online* $\mathrm{CD^2TSP}(1,2)$ *is* $5/3$ *and the greedy algorithm is optimal.*

We give two lemmas that together give us the proof of Theorem 8.

Lemma 10. *The competitive ratio of any online algorithm for* $\mathrm{CD^2TSP}(1,2)$ *with two time zones is at least* $5/3$.

Proof. Let $z_1 = n/3$ and assume that each edge in time zone one has cost one. Take an arbitrary on-line algorithm ALG for this problem and consider its performance. The algorithm must produce an initial path of cost and length $n/3$ before acquiring the cost of the edges in time zone two. The adversary makes sure that ZIG[1] and ALG[1] are disjoint and that all edges incident to black vertices get cost one in time zone two. All other edges get cost two. Since the only edges with cost one in time zone two are those incident to vertices already visited by ALG, the cost of time zone two is $4n/3$ and the total cost of the tour is $5n/3$. ZIG can use all cost one edges in time zone two and achieve the total cost n, since ZIG[1] and ALG[1] are disjoint. $\square$

Below, we present an optimal exponential-time online algorithm for CD^{2}TSP$(1,2)$ with two time zones. The input to the algorithm is the cost matrix for time zone one and at time z_1 the algorithm is given the cost function of time zone two.

```
Algorithm   $A_{exp}$
1  Compute the longest path of cost $\leq z_1$
2  Get new input (new cost function)
3  Compute the min cost path from the current position to the starting point.
End  $A_{exp}$
```

Lemma 11. *A_{exp} has competitive ratio* $5/3$.

Proof. Let m_1 be the number of edges in the tour produced by A_{exp}, restricted to time zone one, and $A_{exp}(I_2)$ the cost of the tour restricted to time zone two. Furthermore, let m_2 and $\mathrm{OPT}(I_2)$ be the corresponding variables for the optimal tour. The competitive ratio of A_{exp} can be expressed as

$$R(A_{exp}) = \frac{z_1 + A_{exp}(I_2)}{z_1 + \mathrm{OPT}(I_2)}.$$

The cost one edges that can be used by an offline algorithm but not by the online algorithm are the edges in time zone two that are incident to black vertices. The maximal number of such edges included in a tour is $\min\{2m_1, n-m_2\}$. The rest of the edges used by the offline algorithm in the second time zone may also be used by A_{exp}. These edges connect a number of vertices that both tours visit in this time zone. If we let *opt* denote the shortest TSP-path among these vertices, we get that $\mathrm{OPT}(I_2) \geq \min\{2m_1, n-m_2\} + \mathit{opt}$.

Let us apply the same argument for the cost two edges in time zone two. The cost two edges that A_{exp}'s tour may be forced to visit but that the optimal tour may escape are the edges that are adjacent to vertices already visited by the optimal tour. The maximal number of such edges in A_{exp}'s tour is $\min\{2m_2, n-m_1\}$, and since A_{exp} computes an optimal TSP-path in time zone two, it follows that $A_{exp}(I_2) \leq 2\min\{2m_2, n-m_1\} + \mathit{opt}$. Since $m_2 \leq m_1$ it follows that the competitive ratio is

$$R(A_{exp}) \leq \frac{z_1 + 2\min\{2m_1, n-m_1\} + \mathit{opt}}{z_1 + \min\{2m_1, n-m_1\} + \mathit{opt}} = 1 + \frac{\min\{2m_1, n-m_1\}}{z_1 + \min\{2m_1, n-m_1\} + \mathit{opt}}.$$

This ratio is maximized for $m_1 = n/3$ for which $\mathit{opt} = 0$, and since $z_1 \geq m_1$, we get that

$$R(A_{exp}) \leq 1 + \frac{2n/3}{n/3 + 2n/3} = \frac{5}{3}.$$

□

Using the improved greedy algorithm for CD^{2}TSP we can achieve the same approximation ratio in polynomial time.

Corollary 3. *There is a polynomial time offline* $5/3$*-approximation algorithm for* CD^{2}TSP$(1,2)$.

Note that we achieve the same result as for the two time zone SDTSP. However, because of the increased complexity in the dependency between time and cost we are not able to generalize the result to hold for k time zones.

6 Conclusions

We study online strategies for two versions of the time dependent traveling salesman problem (SD^{k}TSP and CD^{k}TSP).

For the online version of SD^{k}TSP we achieve an optimal exponential time strategy with competitive ratio $(\frac{M}{m}-1)\frac{2^k-2}{2^k-1}+1$, where M is the largest and m the smallest edge cost, and k is the number of time zones.

In order to make the online strategy time efficient we study the orienteering problem, which appears as a subproblem in the online strategy. We find an approximation algorithm for the orienteering problem with approximation ratio 3/4 if the edge costs are restricted to one and two.

Using the online result for SD^{k}TSP together with the approximation algorithm for the orienteering problem we are able to produce a greedy approximation algorithm with approximation factor $2-\frac{2}{3k}$ for graphs with edge costs one or two. We also give similar results for CD^{2}TSP$(1,2)$, matching those found for SD^{2}TSP$(1,2)$.

References

1. E. M. Arkin, G. Narasimhan, and J. S. B. Mitchell. Resource-constrained geometric network optimization. In *Proc. Fourteenth ACM Symposium on Computational Geometry*, pages 307–316, 1998.
2. G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Algorithms for the on-line traveling salesman. *Algorithmica*, 29(4):560–581, 2001.
3. N. Balakrishnan, A. Lucena, and R. T. Wong. Scheduling examinations to reduce second-order conflicts. *Computers in Operations Research*, 19:353–361, 1992.
4. Michiel Blom, Sven O. Krumke, Willem de Paepe, and Leen Stougie. The online-TSP against fair adversaries. *Lecture Notes in Computer Science*, 1767:137–149, 2000.
5. K. Fox. *Production Scheduling on Parallel Lines with Dependencies.* PhD thesis, The John Hopkins University, Baltimore, MD, 1973.
6. K. Fox, B. Gavish, and S. Graves. An n-constraint formulation of the (time-dependent) traveling salesman problem. *Operations Research*, 28:1018–1021, 1980.
7. H. N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *Proc. of the 1st annual ACM-SIAM Symposium on Discrete Algorithms*, pages 434–443, 1990.
8. M. R. Garey, R. L. Graham, and D. S. Johnson. Some NP-complete geometric problems. In *Proc. 8th Annual ACM Symposium on Theory of Computing*, pages 10–21, 1976.
9. M. Hammar and B.J. Nilsson. Approximation results for kinetic variants of tsp. *Discrete and Computational Geometry*, 27(4):635–651, 2002.
10. C. S. Helvig, G. Robins, and A. Zelikovsky. Moving-target TSP and related problems. In *Proc. 6th Annual European Symposium on Algorithms (ESA)*, 1998.
11. M. Lipmann. The online traveling salesman problem on the line. Master's thesis, Department of Operations Research, University of Amsterdam, The Netherlands, 1999.
12. C. Malandraki and M. S. Daskin. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science*, 26:185–200, 1992.
13. C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18(1):1–11, 1993.
14. J. C. Picard and M. Queyranne. The time-dependent traveling salesman problem and its application to the tardiness problem in one machine scheduling. *Operations Research*, 26:86–110, 1978.