Approximation Algorithms for Time-Window TSP and Prize Collecting TSP Problems

Jie Gao[1], Su Jia[1], Joseph S. B. Mitchell[1], and Lu Zhao[1]

Stony Brook University, Stony Brook, NY 11794, USA.
{jie.gao, su.jia, joseph.mitchell, lu.zhao}@stonybrook.edu.

Abstract. We give new approximation algorithms for robot routing problems that are variants of the classical traveling salesperson problem (TSP). We are to find a path for a robot, moving at speed at most s, to visit a set $V = \{v_1, \ldots, v_n\}$ of sites, each having an associated time window of availability, $[r_i, d_i]$, between a release time r_i and a deadline d_i. In the *time-window prize collecting problem (TWPC)*, the objective is to maximize the number of sites visited within their time windows. In the *time-window TSP problem (TWTSP)*, the objective is to minimize the length of a path that visits *all* of the sites V within their respective time windows, if it is possible to do so within the speed bound s. For sites on a line, we give approximation algorithms for TWPC and TWTSP that produce paths that visit sites v_i at times within the relaxed time windows $[r_i - \varepsilon L_i, d_i + \varepsilon L_i]$, for fixed $\varepsilon > 0$, where $L_i = d_i - r_i$; the running time is $O((nL_{max})^{O(\frac{\log L_{max}}{\log(1+\varepsilon)})})$, where $L_{max} = \max_i L_i$. For TWPC, the computed path visits at least k^* (the cardinality of an optimal solution to TWPC) sites; for TWTSP, the computed path is of length at most λ^* (the length of an optimal TWTSP solution). For general instances of sites in a metric space, we give approximation algorithms that apply to instances with certain special structure of the time windows (that they are "dyadic" or that they are "elementary"), giving paths whose lengths are within a bounded factor of the optimal length, $\lambda^*(s)$, for the given speed s, while relaxing the speed to be a factor greater than s; for arbitrary time windows, we give an $O(\log n)$-approximation for TWTSP, assuming unbounded speed ($s = \infty$).

1 Introduction

Advances in mobile robotics and autonomous vehicles have given rise to a variety of new applications and research challenges. As the hardware and control systems have matured, a number of new strategic planning problems have emerged as robots and autonomous vehicles become deployed in our living spaces, urban areas, and roadways. Our work is motivated by the scheduling and motion planning problem when the tasks that need to be done are associated with both locations and with time windows. For example, an autonomous vehicle may be tasked with performing package pickup/delivery operations, each with a physical location and a window of time during which the pickup/delivery is expected to

take place. Our goal, then, is to find an efficient path to pick up or deliver all of the packages (or as many as possible), subject to the given time windows.

The classical *travelling salesperson problem (TSP)* seeks a shortest path or cycle to visit a set $V = \{v_1, \ldots, v_n\}$ of n *sites* in a metric space (e.g., the Euclidean plane); the challenge is to determine the order in which to visit the sites. In this paper we study two variants of the TSP. In each variant, we assume that there is a single mobile robot, which can move at a maximum speed s. The robot is initially located (at time $t = 0$) at location v_0; the location v_0 might be given, as a fixed *depot*, or might be flexible, allowing us to determine the best choice for v_0. The input to our problems includes, for each site $v_i \in V$, a specified *time window*, $[r_i, d_i]$, with *release time* r_i and *deadline* d_i, during which the site v_i is to be visited. (A further generalization of our problems includes a *processing time* associated with each site v_i, indicating the amount of time that must be spent at v_i; we assume here that processing times are zero.)

In the *time-window prize collecting problem (TWPC)*, the objective is to determine a path P for the robot that maximizes the number of sites (or, more generally, the sum of "prizes" associated with sites) visited within their time windows; we let k^* denote the number of sites visited by an optimal TWPC path, P^*.

In the *time-window TSP problem (TWTSP)*, the objective is to determine a path P of minimum length that visits *all* of the sites V within their respective time windows, if it is possible to do so within the speed bound s; we let λ^* denote the length of an optimal TWTSP path P^*.

Related Work. Both the TWPC and TWTSP problems are NP-hard, in general, since they generalize the classic TSP. The TSP has been studied extensively (see, e.g., [11]), and polynomial-time approximation schemes are known for geometric instances (see, e.g., [2, 12, 13]). In 1D (i.e., for points on a line), the TSP is trivial. However, the TWTSP and TWPC problems are known to be strongly NP-complete even in 1D [15]. Bockenhauer *et al.* [7] showed that there is no polytime constant-factor approximation algorithm for TWTSP in metric spaces, unless P = NP.

Approximation algorithms for the time window prize collecting (TWPC) problem have been studied. Bar-Yehuda *et al.* [5] gave an $O(\log n)$-approximation algorithm for n sites on a line. For general metric spaces, Bansal *et al.* [4] gave an an $O(\log^2 n)$-approximation algorithm in general and an $O(\log n)$-approximation for the special case with release times $r_i = 0$. Chekuri *et al.* [8] gave an algorithm with approximation $O(\text{poly}(\log \frac{L_{max}}{L_{min}}))$, where L_{max} and L_{min} are the maximum and minimum lengths of the time windows. The online version was recently studied by Azar *et al.* [3]. The special case of the TWPC problem in which all release times are zero ($r_i = 0$) and all deadlines are the same ($d_i = d$, for all i) is known as the *orienteering problem*; the objective is to visit as many sites as possible with a path of length at most d/s. Approximation algorithms are known for orienteering, including polynomial-time approximation schemes for geometric instances (see, e.g., [1, 6, 9, 14]).

The TWTSP has also been studied in the operations research literature, using integer programming and branch-and-bound techniques; see [10] for a survey. These algorithms are not reviewed here, as our emphasis is on provable approximation algorithms that are polynomial-time (or potentially quasipolynomial-time).

Preliminaries. The input set of n sites $V = \{v_1, \ldots, v_n\}$ lie in a metric space; we let $\delta(v_i, v_j)$ denote the distance between v_i and v_j. We assume that time windows $[r_i, d_i]$ associated with the sites v_i have release times r_i and deadlines d_i, with $r_i, d_i \in [0, T]$ for time horizon T. We let $L_i = d_i - r_i$ denote the length of the time window associated with site v_i, and we let $L_{max} = \max_i L_i$ be the length of the largest time window.

A time window $[r_i, d_i]$ is *dyadic* if $L_i = 2^m$, for some integer $m \geq 0$, and r_i is an integer multiple of L_i. We say that an input is a *dyadic instance* if all time windows are dyadic. We say that an input is an *elementary instance* if (1) each time window is either of unit length (i.e., $d_i = r_i + 1$) or is of full length, with $[r_i, d_i] = [0, T]$, for integer T; and (2) for each integer $j \in [0, T-1]$, there exists at least one site having time window $[j, j+1]$.

Our Contributions. We provide a collection of new results for the TWPC and TWTSP, including:

1. For points V on a line (i.e., 1D), we provide dual approximation algorithms, running in time $O((nL_{max})^{O(\frac{\log L_{max}}{\log(1+\varepsilon)})})$ for both the TWPC and TWTSP problem for any fixed speed bound s. In other words, we find a path that performs as well as the optimal, but that allows each site v_i to be visited in the relaxed time window $[r_i - \varepsilon L_i, d_i + \varepsilon L_i]$, where $L_i = d_i - r_i$.
   Our method also provides an approximation for TWPC in 1D when only relaxation of the deadline is allowed, computing a path to visit $\geq k^*$ sites, allowing each site to be visited in the time window $[r_i, (1+\varepsilon) \cdot d_i]$. This improves the results by Bansal *et al.* (in [4]), which found in polynomial time a path to visit $\Omega(\frac{k^*}{\log 1/\varepsilon})$ sites with the same relaxation on time windows.
2. As a byproduct of our method, we give new approximation algorithms for the Monotone TSP with Neighborhoods (TSPN) and Monotone Orienteering with Neighborhoods problem in 2D for arbitrary regions (arbitrary size, overlapping and fatness).
3. For TWTSP with finite speed s in a metric space, we present an (α, β) dual approximation algorithm, using speed $\leq \alpha \cdot s$ and travel distance $\leq \beta \cdot \lambda^*(s)$, where $\lambda^*(s)$ is the length of an optimal path subject to speed bound s, $\alpha, \beta = O(1)$ for an elementary instance, and $\alpha, \beta = O(\log L_{max})$ for a dyadic instance. For $s = \infty$, we give an $O(\log n)$-approximation for arbitrary time windows.

While the TWPC problem is well studied (for over 20 years), the best known factor for TWPC in 1D is still $O(\log L_{max})$ or $O(\log n)$ (as it is in metric spaces), if we strictly insist on visiting points in their time windows. Little attention has been given to dual approximations. Dual approximation schemes are natural approaches to addressing hard optimization problems. Further, relaxation of

time windows is realistic, since the release times and deadlines often have some flexibility. Our results show that if we are allowed to relax the time windows even by a little bit, we can obtain much better approximation. Our results also reveal that the time window variants of TSP in 1D are significantly different from the problems in higher dimensions or in metric spaces.

For the TWTSP problem in a metric space, one may ask whether it is possible to apply the known method for TWPC. The difficulty is as follows. Recall the strategy in [4, 5]: first, classify the vertices into $g = O(\log n)$ or $O(\log^2 n)$ groups according to their time windows, such that the time windows in each group are roughly the same. Then, note that when all time windows are the same, the TWPC problem is just the ordinary prize collecting problem, so we can find a constant-factor approximation for each group. Among these g solutions, choose the one with the largest prize. A natural idea to apply to the TWTSP problem is to find a constant-factor approximation for each group and then to paste them together in an appropriate way. This strategy works for TWTSP with infinite speeds but fails when the speed is bounded, since it does not give any guaranteed bounds in speed, and consequently we may return a solution with speed much higher than s. We explain briefly the techniques we use in this paper.

Overview of Our Approach. We view the 1D problem in space-time, in the (t, x) plane, where t is the (horizontal) time coordinate and x is the (vertical) spatial coordinate along the line containing the sites V. Then, the sites $v_i \in V$ are points along the (vertical) x-axis, and the time windows $[r_i, d_i]$ are horizontal line segments σ_i in the (t, x) plane. A path P corresponds to a slope-bounded piecewise-linear function, $P(t)$, with absolute value of slope at most s. Visiting a site v_i within its time window corresponds to the path $P(t)$ visiting the corresponding (horizontal) line segment σ_i.

To get our results for 1D problems, we first look at special cases called *dyadic instances* when all time windows are *dyadic*. These dyadic intervals can be considered as intervals of a binary recursive partition. We can run dynamic programming to find the best path. Then we introduce the h-dyadic instance, which, intuitively can be solved by partitioning each dyadic interval at at most h places called partition points. Our method consists of the following steps: (1) we give an $O((nL_{max})^{O(h)})$-time algorithm for h-dyadic instances; (2) we show how arbitrary time windows can be expanded to have endpoints among a set of h carefully selected partition points placed within dyadic intervals, thereby transforming a general instance into an h-dyadic instance. We solve h-dyadic instances using a carefully designed dynamic programming algorithm, taking advantage of the fact that dyadic intervals have a hierarchical structure. Within each subinterval of a dyadic interval, the subproblems of our dynamic program keep track of the x-extent (min-x and max-x) of a solution path.

Step (2) may appear to be straightforward, but there are two challenges in assigning partition points. One is the tradeoff between the precision of our relaxation and the number of partition points: if the partition points are too dense, then we end up with a high running time; if the partition points are too sparse, then the relaxation of time windows is too coarse. The other difficulty is

that the partitions for dyadic intervals are not independent; they are correlated, in the sense that the partition of any dyadic interval must inherit all the partition points from its parent.

For the dual approximation for TWTSP in metric spaces on elementary instances, we first group the sites so that the weight of the minimum spanning tree (MST) of V restricted to each group is $O(\log L)\lambda^*(s)$. Then, we solve a matching problem on a bipartite graph whose "red" nodes correspond to sites having unit-length time windows and whose "blue" nodes correspond to the groups, allowing us to assign the groups to red nodes in a balanced manner, thereby avoiding the need for the speed to be increased by more than a particular bound.

2 TWTSP and TWPC on a Line

We start with the case when the nodes are on a line (or on a curve). We first look at the special case when the robot may travel with infinite speed. For this case, the problem of visiting all sites within their time windows is always feasible. The goal therefore is to minimize the travel distance. We show a dual algorithm for this instance as below.

Theorem 1. *Given an instance for 1D TWTSP problem when the speed bound s for the robot is ∞, let L_{max} be the maximum length of the input segments. Then for any $\varepsilon > 0$, in $O(n^{O(\frac{\log L_{max}}{\log(1+\varepsilon)}} \log L_{max})$ time we can find a path P, such that*

1. *the length of P is at most OPT, the optimum of the problem,*
2. *each segment σ_i is visited in time window $[r_i - \varepsilon L_i, d_i + \varepsilon L_i]$, where $L_i = d_i - r_i$.*

To prove this theorem we first look at special cases called *dyadic instances* when all time windows are *dyadic*. That is, when the length of each time window is a power of two, and the release time is a nonnegative integer multiple of its length. For a dyadic interval I, let I_L and I_R be its the left and right child interval respectively, when we cut the interval I at its midpoint. In Subsection 2.1 we first give a polytime algorithm for dyadic instances using dynamic programing.

In Subsection 2.2 we generalize the above algorithm to an $O(n^{O(h)} \log L_max)$ time algorithm for *h-dyadic instances*, which is defined below. Consider a dyadic interval $I = [a, b]$ of the t-axis, and let $S(I)$ be the segments fully contained in the slab $I \times [-\infty, \infty]$ and stabbed by the midline of this slab, i.e. $\{(a + b)/2\} \times [-\infty, \infty]$. We call an instance *$h$-dyadic*, if we can partition each dyadic interval I at interger values (called the partition points) into at most h pieces so that (1) for each segment s in $S(I)$, the two endpoints project to partition points on the t axis; and (2) for every dyadic interval $I = [a, b]$, every partition point for I is also a partition point for the children intervals of I, i.e. $[a, \frac{1}{2}(a + b)]$ and $[\frac{1}{2}(a + b), b]$, but not vice-versa.

Last we show that for any $\varepsilon > 0$, we can transform any instance $\mathcal{I}$ to an $O(\frac{\log L_{max}}{\log(1+\varepsilon)})$-dyadic instance $\mathcal{I}'$, such that each time window is stretched by at most $(1 + \varepsilon)$ times (Subsection 2.3), which completes the proof.

After we have a good understanding of the case when the robot can travel with infinite speed, we now handle the general case when the robot speed is bounded by s.

Theorem 2. *Given an instance for 1D TWPC problem with bounded velocity s, let L_{max} be the maximum length of the input segments, and assume the shortest time window has length ≥ 1. Then for any $\epsilon > 0$, in $O((nL_{max})^{O(\frac{\log L_{max}}{\log(1+\varepsilon)})})$ time we can find a path P, such that*

1. *the number of segments that P visits is at least OPT,*
2. *each segment σ_i is visited in $[r_i - \varepsilon L_i, d_i + \varepsilon L_i]$, where $L_i = d_i - r_i$.*

Similar result holds for 1D TWTSP with finite speed.

2.1 Infinite Speed ($s = \infty$) for Dyadic Instance

Lemma 1. *For dyadic instances, the 1D TWTSP problem with infinite speed ($s = \infty$) can be solved in poly(n) time.*

Proof. The main idea is to use dynamic programming. For that, we now introduce the subproblems. Let $|P|$ be the length of a path P. Given a dyadic interval $I = [a, b]$, we use $P^* = OPT(I; \theta)$, where $\theta = (x_N, x_S, x_b, x_e)$, to denote the optimum of the following subproblem:

Minimize $|P|$, s.t.

1. P visits all segments that are fully contained in I;
2. the points with maximum and minimum x-coordinate that P visits in I are x_N and x_S respectively;
3. P starts and ends in x_b and x_e respectively.

If there is no feasible solution to the problem of $OPT(I; \theta)$, then we take its value to be $-\infty$. This happens when the parameters in θ are contradictory to each other. For example, $x_N < x_S$, or $x_b > x_N$ or $x_e < x_S$, we do not enumerate them here.

Now define $P^*|_{I_L}$ and $P^*|_{I_R}$ as the restriction of P on the two children intervals I_L, I_R (that is, I is partitioned in the middle and the left one is I_L and the right one is I_R). By an exchange argument, $P^*|_{I_L}$ is in fact the optimal of $OPT(I_L; \theta_L^*)$, where θ_L^* is the parameter induced by P_L^*: θ_L^*=($x_{N,L}$, $x_{S,L}$, $x_{b,L}$, $x_{e,L}$), where $x_{N,L}/x_{S,L}$ is the max/min x-coordinates position that P^* visits in I_L and $x_{b,L}/x_{e,L}$ is the starting point/ending point of P^* in I_L. Specifically, if there is another path P' on I_L having parameter θ_L^* whose length is shorter than $P^*|_{I_L}$, then the concatenation $P' \cup P^*|_{I_R}$ should be shorter than P^*. That is a contradiction. Similarly, we know $P^*|_{I_R} = OPT(I_R; \theta_R^*)$.

Given $\theta, \theta_L, \theta_R$, where $\theta_L = (x_{N,L}, x_{S,L}, x_{b,L}, x_{e,L})$, and $\theta_R = (x_{N,R}, x_{S,R}, x_{b,R}, x_{e,R})$, we say θ_L, θ_R are *compatible* with θ, if $\max\{x_{N,L}, x_{N,R}\} = x_N$, $\min\{x_{S,L}, x_{S,R}\} = x_S$, and $x_{e,L} = x_{b,R}$. Hence, we have the following recurrence: $OPT(I; \theta) = \min\{OPT(I_L; \theta_L) + OPT(I_R; \theta_R)$: θ_L, θ_R are compatible with $\theta\}$.

Therefore, if we know $OPT(J;\theta)$ for each Level(k) interval J and all θ, then we are able to compute $OPT(I;\theta')$ for all Level($k+1$) interval and all θ' in $O(n^{12}L_{max}\log L_{max})$ time. Specifically, for a fixed θ, we find the minimum value over $O(n^4)\cdot O(n^4) = O(n^8)$ choices of θ_L and θ_R. Since there are $O(n^4)$ different θ, it takes $O(n^{12})$ to find $OPT(I;\theta)$ for all θ. Since there are $O(\log L_{max})$ rows in our lookup table, each row with $O(L_{max})$ dyadic intervals, the total running time for computing all subproblems is $O(n^{12}L_{max}\log L_{max})$. Now by rescaling the t-axis L_{max} can be viewed as linear in n. So the running time is $O(n^{13}\log n)$.

2.2 Infinite Speed ($s = \infty$) for h-dyadic Instance

To generalize the dynamic programming idea used in the dyadic instance, we consider the notion of an *h-dyadic* instance by capturing the intuition that the solution for a larger interval can be solved by using solutions for h subintervals. To do that, we first explain the definitions of h-dyadic instances.

Given an interval $[a,b]$, let π: $a = t_0 \le t_1 \le ... \le t_{k-1} \le t_k = b$ be a partition of $[a,b]$ into $k \le h$ subintervals, we call such a partition an *h-partition*, and each t_i is called a partition point. Now we define the *Inheriting Property*. Given a collection of dyadic intervals each associated with a h-partition, $\pi(I)$ for the dyadic interval I, we say this family of partitions has the inheriting property, if for any two sibling dyadic intervals I_1, I_2 (i.e., whose union/parent I is also a dyadic interval), the union of the partitioning points of $\pi(I_1)$, $\pi(I_2)$ is a superset of the partitioning points of $\pi(I)$.

An instance for TWTSP is called *h-dyadic*, if we can associate an h-partition $\pi(I)$ to each dyadic interval I, such that this family of partitions has the inheriting property, and for each input segment σ_i in the instance, both endpoints of σ_i are partition points of $\pi(W(\sigma_i))$, where $W(I)$ as the minimal dyadic interval containing I.

Now we modify the dynamic programming for dyadic instances to h-dyadic instances. We need the following subproblem definition (See Figure 2.2).

Definition 1. *(Subproblem for h-dyadic instance) Let $\mathcal{I}$ be an instance to 1D TWTSP problem with infinite speed bound. Let π be an h-partition of $I = [a,b]$, with partition points $\{t_i\}$. Define $OPT(I;\pi;\theta)$, where $\theta = (x_N^1,..., x_N^h; x_S^1,...,x_S^h; x_b, x_e)$, as the optimum of the following problem:*

Minimize $|P|$, s.t.

1. *P visits all segments that are fully contained in I;*
2. *the maximum and minimum x-coordinates that P visits in $[t_{j-1}, t_j]$ are x_S^j and x_N^j respectively, denoted as the vertical range $[x_S^j, x_N^j]$;*
3. *P starts and ends in x_b and x_e respectively.*

Let $t_i \le t_j$, define $V[t_i, t_j]$ as the set of segments whose projection to the t-axis is *exactly* $[t_i, t_j]$. Here is a crucial observation (refer to Fig 2.2): given a dyadic interval $I = [a,b]$, associated with an h-partition $\pi : t_0 = a \le t_1 \le ... \le t_h = b$. Then, path P visits all the segments in $V[t_i, t_j]$ *if and only if*

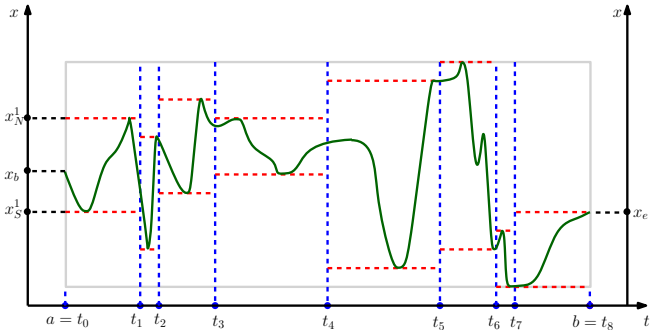

Fig. 1. Illustration of $OPT(I;\pi;\theta)$. The red dashed segments are the max/min x-coordinates P visits in each interval $[t_{i-1}, t_i]$. The points x_N^1, x_S^1, x_b, x_e are highlighted.

the maximum x-coordinates that P visits in the interval $[t_i, t_j]$ is "higher" than the "highest" in $V[t_i, t_j]$, and minimum x-coordinates that P visits in $[t_i, t_j]$ is "lower" than the "lowest" segment in $V[t_i, t_j]$.

Theorem 3. *The 1D TWTSP with $s = \infty$ for h-dyadic instances can be solved in $O(n^{O(h)} \log L_{max})$ time.*

Proof. The dynamic programming algorithm differs from the dyadic case in two ways. First, we have a more complicated parameter θ, which now encodes the vertical ranges in *every* subinterval, hence having $O(h)$ length. Second, we have a slightly different recursive structure: in addition to the requirement that θ_L, θ_R be compatible with θ, we also require that all non-dyadic segments in I be visited.

Let π, π_L, π_R be the h-partitions associated with dyadic intervals I, I_L, I_R respectively, where I is the parent of I_L, I_R. Given a segment σ parallel to t-axis, let $x(\sigma)$ be its x-coordinate. We have the following recursive relation:

$$OPT(I;\pi;\theta) = \min_{\theta_L,\theta_R} \{OPT(I_L;\pi_L;\theta_L) + OPT(I_R;\pi_R;\theta_R)\},$$

for any $j \le h$,

$$x_N^j = \max\{x_N^i : t_{j-1} \le t_{i-1}^L \le t_i^L \le t_j \text{ or } t_{j-1} \le t_{i-1}^R \le t_i^R \le t_j\},$$

$$x_S^j = \min\{x_S^i : t_{j-1} \le t_{i-1}^L \le t_i^L \le t_j \text{ or } t_{j-1} \le t_{i-1}^R \le t_i^R \le t_j\},$$

and for any pair $i, j \le h$,

$$\max\{x_{N,L}^i, ...x_{N,L}^h, x_{N,R}^1, ..., x_{N,R}^j\} \ge \max\{x(\sigma) : \sigma \in V[t_i, t_j]\},$$

$$\min\{x_{S,L}^i, ...x_{S,L}^h, x_{S,R}^1, ..., x_{S,R}^j\} \le \min\{x(\sigma) : \sigma \in V[t_i, t_j]\}.$$

There are four constraints in the relation above. The first two say that if the max/minimum x-coordinates that P visits in $[t_i, t_j]$ is x, then P visits x in at least one subinterval of π_L or π_R contained in $[t_i, t_j]$. The last two constraints are saying that P must visit all segments in $\mathcal{S}(I)$. Indeed, for each i, j, the segments in $V[t_i, t_j]$ are all visited by P *if and only if* the union of vertical ranges of P in $[t_i, t_j]$ fully contains the vertical range of $V[t_i, t_j]$, i.e. $[S, N] \subset \cup_{i \le k \le j}[x_S^k, x_N^k]$, where $S = \min\{x(s) : s \in V[t_i, t_j]\}$, $N = \max\{x(s) : s \in V[t_i, t_j]\}$.

Since there are $O(n^{O(h)} \log L_{max})$ entries in the lookup table, the proof is complete.

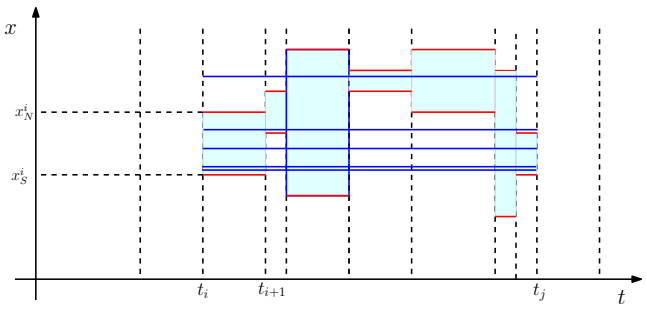

Fig. 2. Illustration of Theorem 3. Segments in $V[t_i, t_j]$ are all visited by P *if and only if* the union of vertical ranges of P in $[t_i, t_j]$ fully contains the vertical range of $V[t_i, t_j]$, For example, all segments in $V[t_i, t_j]$ (colored blue) are all visited by P.

2.3 Infinite Speed ($s = \infty$) for General Case and Generalizations

For a general instance, how do we approximate a general instance using an h-dyadic instance? Our idea is as follows: associate a partition to each dyadic interval, and then stretch the endpoints of each segment σ to some partition points of $\pi(W(\sigma))$, where $W(\sigma)$ is the minimal dyadic interval containing σ.

But how to find these partitions? The challenge is as follows: on one hand, we want the partition points to be sparse, so that we have fewer entries in our dynamic programming table and hence having a small running time; on the other hand, we want them to be dense, so that we can stretch each segment by a factor of at most $1 + \varepsilon$. Moreover, we wish this family of partitions to have a nice structure – the "inheriting property" – so that we can use dynamic programming. Below we define this partition formally.

Given an integer interval $I = [a, b]$, let $l = b - a$ and $c = \frac{1}{2}(a + b)$. We say a partition π is a *symmetric two-sided logarithmic ε-dense partition* (in short *ε-dense* partition) of I if (1) all partition points are integers, (2) this partition is symmetric with respect to the midpoint c, (3) for any integer $q \le \log_{1+\varepsilon} l$, the subinterval $[(1 + \varepsilon)^q + a, (1 + \varepsilon)^{q+1} + a]$ contains at least one partition point, unless it contains no integers at all.

The following shows that such a family of partitions exists. The proof is omitted in this version.

Lemma 2. *(ε-dense Partition with Inheriting Property) Let $L_{max} > 0$ be a power of 2. We can assign a partition $\pi(I)$ to each dyadic interval $I \subseteq [0, L_{max}]$, such that*

1. *$\{\pi(I)$: dyadic $I \subseteq [0, L_{max}]\}$ is a family of partitions with the inheriting property,*
2. *each $\pi(I)$ has at most $O(\frac{\log L_{max}}{\log(1+\varepsilon)})$ partition points,*
3. *$\pi(I)$ is ε-dense for each dyadic interval $I \subseteq [0, L_{max}]$.*

Finally, we show our dual approximation for 1D TWTSP with infinite speed.

Proof (Theorem 1). First we construct a family of ε-dense partition Π. Then, transform our instance into an h-dyadic instance $\mathcal{I}'$, where $h = O(\frac{\log L_{max}}{\log(1+\varepsilon)})$, by

stretching each segment σ_i so that both its endpoints are the partition points of $\pi(W(\sigma_i))$. Find the optimal solution P for $\mathcal{I}'$ using the dynamic programming. Clearly $|P| \leq OPT$. Note that we have stretched each segment by at most $1+\varepsilon$ times, so P visits each segment σ_i in time interval $[r_i - \varepsilon L_i, d_i + \varepsilon L_i]$, and the proof is complete.

2.4 Bounded Speed ($s < \infty$)

The generalization of the above algorithm to a finite speed scenario is non-trivial. First the problem may not admit a feasible solution if the robot speed is too slow. Therefore we look at the prize-collecting problem (TWPC). Second, it is possible that we need to sacrifice some immediate interest in order to obtain more long-term interest. So the solution structure might completely change when the speed cap is placed. We will overcome this by carefully defining subproblems and encoding more parameters in our dynamic programming.

Without loss of generality, we consider rectilinear paths, travelling at most distance 1 in unit time. For simplicity, we assume the prize placed at each point is one, though the proof can be easily generalized to the case where arbitrary prize are allowed at each point.

Let $\mathcal{S}$ be a set of segments parallel to the t-axis, and $I = [a, b]$ be any integer interval. Let $\mathcal{S}(I)$ be the set of segments whose projection on the t-axis is fully inside I. Given path P, let y_N and y_S denote the maximum and minimum coordinates among the segments P visits in $\mathcal{S}([a, b])$, define $B(P, a, b)$ as the rectangle $[a, b] \times [y_S, y_N]$. For an axis-parallel rectangle/box B in the plane, denote the upper/lower boundary as $\partial^N(B)$ and $\partial^S(B)$ respectively. Given a t-monotone path P, let $P(\tau)$ denote the x-coordinate of P at time τ.

Definition 2. *Given a dyadic interval $I = [a, b]$, numbers $\tau_N^-, \tau_N^+, \tau_S^-, \tau_S^+$ and τ_b, τ_e, define $OPT(I; \theta)$, where $\theta = (y_N, y_S; x_N, x_S; x_b, x_e; \tau_b, \tau_e, \tau_N^-, \tau_N^+, \tau_S^-, \tau_S^+)$, as the optimal solution to the following problem.*

Maximize number of segments in $\mathcal{S}(I)$ visited by P, s.t.

1. *the maximum and minimum x-coordinates that P visits in I are x_N and x_S respectively,*
2. *among the segments in $\mathcal{S}(I)$ that P visits, the maximum and minimum x-coordinates are y_N and y_S respectively,*
3. *P starts/stops at location x_b and x_e respectively,*
4. *$\tau_b = \min\{\tau \in [a, b] : P(\tau) \in [y_S, y_N]\}$, $\tau_e = \max\{\tau \in [a, b] : P(\tau) \in [y_S, y_N]\}$,*
5. *P arrives at y_N at time τ_N^- and moves to x_N, and comes back to y_N at time τ_N^+; it also arrives at y_S at time τ_S^- and moves to x_S, and comes back to y_S at time τ_S^+, and finally arrives at x_e.*

To understand this, consider a path as a (monotone) rope, we hit 8 nails into the wall to fix the rope. The coordinates of the 8 nails are: (a, x_b), (τ_b, x_b), (τ_N^-, y_N), (τ_N^+, y_N), (τ_N^-, y_S), (τ_S^+, y_S), (τ_e, x_e), (b, x_e). You are allowed to change the shape of the rope, as long as the nails are fixed, and the maximum and

minimum x-coordinate visited are x_N, x_S. Among all such ropes, find the one hitting maximum number of segments in $\mathcal{S}(I)$. See Fig 3.

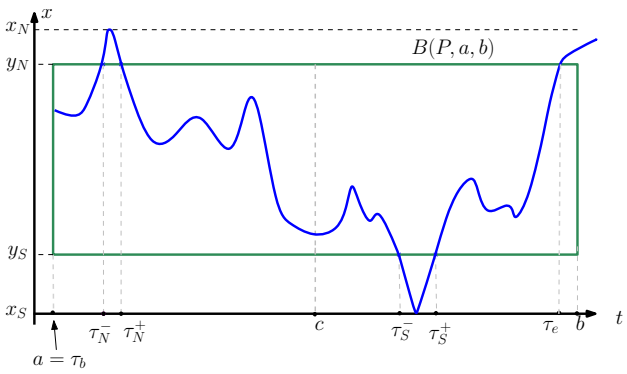

Fig. 3. Illustration of the subproblem $OPT(I;\theta)$. The big green box is $B = B(P,a,b)$.

Now we perform the dynamic programming. Compared with the one in the last section, we have more parameters for finding $OPT(I;\theta)$. We need the maximum and minimum x-coordinate that P visits in I, as well as the coordinates of the highest and lowest segment in $\mathcal{S}(I)$ that P visits, whose position are denoted by y_N and y_S, and the time that P leaves/enters $I \times [y_S, y_N]$.

Define $B := B(P,a,b)$, $B_L := B(P,a,c)$, $B_R := B(P,c,b)$. The following lemma says that a "reasonable" path P crosses the boundary of each of these boxes for $O(1)$ times. With space constraints we omit the proofs in this version.

Lemma 3. *(Polynomial Boundary Complexity of P^*) Let $P^* = OPT(\mathcal{S}([a,b]); \theta)$, and τ_b, τ_e be defined as in Definition 2. Then, P^* leaves/enters each of $\partial^N B_L$, $\partial^S B_L$, $\partial^N B_R$, $\partial^S B_R$ at most once between τ_b, τ_e.*

Corollary 1. *(Foundation of DP) If $P^* = OPT(\mathcal{S}([a,b]);\theta)$, then there exists parameters θ_L and θ_R, such that $P^*|_{[a,c]} = OPT(\mathcal{S}([a,c]);\ \theta_L)$ and $P^*|_{[c,b]} = OPT(\mathcal{S}([c,b]);\ \theta_R)$.*

Lemma 4. *The 1D TWPC problem with dyadic time windows can be solved in* $\mathrm{poly}(n, L_{max})$ *time.*

As in the case of TWTSP, this dynamic programming can be generalized to h-dyadic instances, and leads to the algorithm in Theorem 2.

2.5 Generalization

This section we discuss generalization of the results to several other scenarios.
TWTSP/TWPC for m-robots. The result for TWTSP and TWPC with infinite/finite speed can be easily generalized when m-robots are used, with running time $O((nL_{max})^{O(m\frac{\log L_{max}}{\log(1+\varepsilon)})})$.

Monotone TSPN. Our results can be used to show new results for traveling salesman problem with neighborhood when the path needs to be monotone. Given a set of regions $\{Q_i\}_{i=1,\ldots,n}$ in the plane, we wish to find a horizontally monotone TSPN path which minimizes the vertical distance travelled. Suppose that the width of any region at least 1, and let L_{max} be the largest width. Then for any $\varepsilon > 0$, in $O(n^{O(\frac{\log L_{max}}{\log(1+\varepsilon)})} \log L_{max})$ time we can find a path P, whose vertical distance travelled is at most λ^*, and $\delta(P, Q_i) \leq \varepsilon \cdot width(Q_i)$ for each Q_i. Note that this theorem holds for regions with arbitrary overlapping, shape, and size.

Improvement of Bansal *et al.*'s approximation ([4]) As a byproduct, we also improve Bansal *et al.*'s approximation ([4]) in 1D, but in $O((nL_{max})^{O(\frac{\log L_{max}}{\log(1+\varepsilon)})})$ time. They gave the following bi-criteria approximation for the TWPC problem in general metric space: in $poly(n)$ time find a path that visits $O(\frac{1}{\log \frac{1}{\varepsilon}} k^*)$ points in $[r_i, (1+\varepsilon)d_i]$.

Theorem 4. *For the 1D TWPC problem, in $O((nL_{max})^{O(\frac{\log L_{max}}{\log(1+\varepsilon)})})$ time, we can find a path P which visits at least k^* segments in $[r_i, (1+\varepsilon)d_i]$.*

To prove this theorem, we need 2 lemmas.

Lemma 5. *Given path P and point v on it, let $t_P(v)$ be the time that P visits v. Suppose $t_P(v) \in [\frac{1}{1+\varepsilon} r(v), (1+\varepsilon)d(v)]$ for each v visited by P for some $\varepsilon < 1$. Then, by slowing down P by $(1+\varepsilon)$, we obtain a path P' such that $t_{P'}(v) \in [r(v), (1+3\varepsilon)d(v)]$ for each v visited by P.*

Proof. Since $t_{P'}(v) = (1+\varepsilon)t_P(v)$, we know $t_{P'}(v) \in [r(v), (1+\varepsilon)^2 d(v)]$. Since $\varepsilon < 1$, we have $(1+\varepsilon)^2 \leq 1 + 3\varepsilon$.

Lemma 6. *For any $\varepsilon > 0$, there exists a family of partitions $\Pi = \{\pi(I)$: interval $I \subseteq [0, L]$ is dyadic$\}$ with the inheriting property, such that*

1. *for any dyadic I, and any integer q with $[(1+\varepsilon)^q, (1+\varepsilon)^{q+1}] \cap I \neq \emptyset$, there is a partition point of $\pi(I)$ in the interval $[(1+\varepsilon)^q, (1+\varepsilon)^{q+1}]$, and*
2. *each $\pi(I)$ has $O(\frac{\log L}{\log(1+\varepsilon)})$ partition points.*

Proof. Consider the sequence $Z = \{\ulcorner (1+\varepsilon)^q \urcorner : q = 1, 2, .. \ulcorner \frac{\log L}{\log(1+\varepsilon)} \urcorner\}$. For each dyadic interval $I = [a, b]$, let $\pi(I) = \{a, b\} \cup Z$. Clearly this family of partitions satisfies the inheriting property. By definition, it satisfies condition (1). To verify (2), we note that Z has $O(\frac{\log L}{\log(1+\varepsilon)})$ partition points, hence the number of partition points in $\pi(I)$ is at most $O(\frac{\log L}{\log(1+\varepsilon)}) + 2 = O(\frac{\log L}{\log(1+\varepsilon)})$.

Proof of Theorem 4. For each s_i, with time window $[r_i, d_i]$, stretch its left endpoint leftwards to the nearest partition point of $\pi(W(s_i))$, and stretch its right endpoint rightwards to the nearest partition point of $\pi(W(s_i))$. Let the new segment be s_i', with time window $[r_i', d_i']$. Then, $r_i' \geq \frac{1}{1+\varepsilon} r_i$ and $d_i' \leq (1+\varepsilon)d_i$. Since the new instance is $O(\frac{\log L_{max}}{\log(1+\varepsilon)})$-dyadic, we can find an optimal solution P^* in $O((nL_{max})^{O(\frac{\log L_{max}}{\log(1+\varepsilon)})})$ time. We complete the proof by slowing down P^* by a factor of $(1+\varepsilon)$.

3 General Metric Space

Given an instance of time window TSP problem in a metric space with metric $\delta(\cdot,\cdot)$. Let s_{min} be the smallest speed s such that there exists a feasible solution. For a given speed bound $s \geq s_{min}$, let $\lambda^*(s)$ be the minimum possible travel distance. For $s \geq s_{min}$, an algorithm is an (α, β) dual approximation if the robot moves with speed $\leq \alpha s$, and travels a total distance $\leq \beta\lambda^*(s)$. The key result for our dual-approximation is the following theorem for dyadic time windows.

Theorem 5. *Under the assumption above, if $s \geq s_{min}$, then there is an algorithm with $(O(\log L), O(\log L))$ dual approximation for the TWTSP problem for dyadic instance, with running time $O(nL_{max} \log n + n^{1.5} L_{max})$.*

When $s = \infty$, we can have the following stronger result.

Theorem 6. *For $s = \infty$ and arbitrary time windows, there is a polytime $O(\log n)$-approximation for the TWTSP problem in a metric space.*

We do not have space to present the proofs for the above cases. But we will present the results and proofs for the most important special case which leads to the results in the general case.

3.1 Proof for an Elementary Case

We begin with a special case called an elementary instance: All release times are integers; Each time window is either unit-length or $[0, L]$, for some fixed integer L; For each $i \leq L$, there is at least one node with time window $[i-1, i]$.

Theorem 7. *For TWTSP problem in general metric space and $s \geq s_{min}$, there is a* poly(n) *time $(O(1), O(1))$ dual approximation for an elementary instance.*

We call the nodes with unit time window "red nodes", and those with time window $[0, L]$ "black nodes". By losing a constant factor in speed, we can assume the path takes the following form (See Fig 4): it starts from a red node with time window $[0, 1]$, say Red_1, then visits some black nodes (the black cycle in Fig 4), then returns to Red_1, and visits all other red nodes with time window $[0, 1]$ (the red cycle in Fig 4), and then goes to a red node with time window $[1, 2]$, say Red_2, and repeat. Hence, among all the nodes with time window $[i, i+1]$, there is a "representative" red node which is attached with two cycles each of length at most s, one black and one red, denoted by $Cycle_B(Red_i)$ and $Cycle_R(Red_i)$ respectively.

We first find an MST of the sites and cut the tree into subtrees called "blocks" $\mathcal{T} = \{T_1,...,T_k\}$, such that (1) all blocks have size between $C_1 s$ and $C_2 s$ except at most one block (called the "exceptional block") whose size is less than $C_1 s$, and (2) every vertex is contained in at most two blocks. For each $i \leq L$, pick an arbitrary red node with time window $[i-1, i]$ as Red_i, build an unweighted bipartite graph H on the chosen red nodes $\{Red_i\}_{1 \leq i \leq L}$ and subtrees $\{T_j\}$: add an edge (Red_i, T_j) if $\delta(T_j, Red_i) \leq s$.

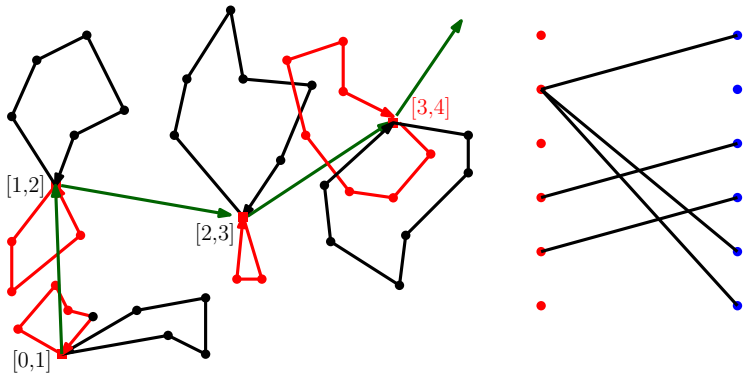

Fig. 4. (a) Left: Each black and red cycle is of length at most s. The distance between Red_i and Red_{i+1} is at most s. (b) Right: A perfect Δ-matching, $\Delta = 3$.

Find a perfect Δ-matching M in H. Here a Δ-matching is a subgraph such that each red node has at most Δ edges in the matching and each subtree has exactly one edge in the matching. See Fig 4(b). If $(Red_i, T_j) \in M$, then we say block T_j is "assigned" to red node Red_i.

For each $i \leq L$, start from Red_i, traverse all the blocks assigned to it, then return to Red_i, then visit all other red nodes with time window $[i-1, i]$ along an $O(1)$-approximate TSP tour on them, and finally add the edge between Red_i and Red_{i+1}.

The proof that the above algorithm achieves what we want comes from the following four lemmas. The proofs of some of them are omitted.

Lemma 7. *(Generalized Hall's Marriage Theorem) In a bipartite graph $G = (V_R \cup V_B, E)$, if for any subset X of V_B, the neighboring vertices set $\Gamma_G(X)$ satisfies $|\Gamma_G(X)| \geq \frac{1}{\Delta}|X|$, then there exists a perfect Δ-matching.*

Lemma 8. *Given a weighted undirected graph $G = (V, E)$, let $T_1, ...T_m$ be some node-disjoint subtrees of $MST(G)$. Let $V' = \cup_{1 \leq i \leq m} V(T_i)$ and G' be the restriction of G on V'. Then $MST(G') \geq \sum_i |T_i|$.*

Lemma 9. *Let $A = \{T_1, ...T_k\} \subseteq \mathcal{T}$, and $V(A)$ be the union of nodes in the blocks in A, i.e. $V(A) = \cup_{i \in A} V(T_i)$. Then the black cycles associated with the red nodes in $\Gamma(A)$ together cover $V(A)$, i.e. $V(A) \subseteq \cup_{r \in \Gamma(A)} BlackCycle(r)$.*

Lemma 10. *Suppose $s \geq s_{min}$, then for any set of blocks $A = \{T_1, ...T_k\} \subseteq \mathcal{T}$, we have $|\Gamma(A)| \geq \mu |A|$, where $\mu = \frac{C_1}{\alpha_{st}(C_2+3)}$.*

Proof. Define $V(A) = \cup_{T_i \in A} V(T_i)$, where $V(T_i)$ are the sites in T_i. The idea is to use $MST(V(A) \cup \Gamma(A))$ as an intermediate quantity to derive an inequality of $|A|$ and $|\Gamma(A)|$.

Assume that for any red node $r \in \Gamma(A)$, there is another red node $r' \in \Gamma(A)$, s.t. $\delta(r, r') \leq (C_2 + 2)s$. We will show how to remove this assumption later.

First we derive an upper bound for $MST(V(A) \cup \Gamma(A))$. Since $s \geq s_{min}$, there is a path P visiting all vertices in their time windows. We will use P to construct a spanning tree for $V(A) \cup \Gamma(A)$. By adding $|\Gamma(A)| - 1$ edges, each

with length $\leq s$, we can connect all the black cycles of the red nodes in $\Gamma(A)$. From Lemma 9, we know that this subgraph visits all nodes in $V(A) \cup \Gamma(A)$. Hence,

$$\begin{aligned} MST((V(A)) \cup \Gamma(A)) &\leq |\Gamma(A)|s + (|\Gamma(A)| - 1) \cdot (C_2 + 2)s \\ &\leq ((C_2 + 3) \cdot |\Gamma(A)| - 1)s. \end{aligned}$$

On the other hand,

$$\begin{aligned} MST((V(A)) \cup \Gamma(A)) &\geq \frac{1}{\alpha_{st}} MST(V(A)) \\ &\geq \frac{1}{\alpha_{st}} \sum_{i=1}^{|A|} |T_i| \geq \frac{C_1 \cdot s}{\alpha_{st}} |A|. \end{aligned}$$

The second last inequality is due to Lemma 8. It follows that

$$\frac{|\Gamma(A)|}{|A|} \geq \mu = \frac{C_1}{\alpha_{st}(C_2 + 3)}.$$

Next we remove the assumption that for any set of red nodes $r \in \Gamma(A)$, there is another red node $r' \in \Gamma(A)$, s.t. $\delta(r, r') \leq (C_2 + 2)s$. Build the following auxiliary graph G': its nodes are all red nodes in $\Gamma(A)$, for each pair of nodes in G', say u, v, add an edge (u, v) if $\delta(u, v) \leq 2C_2 s$. Then, these red nodes are classified into several connected components, say $\{C^i\}_i$. For each i, we have $r_i \geq \mu a_i$, where $r_i = |C^i|$ and $a_i = |\Gamma(C^i)|$. It follows that $|\Gamma(A)| \geq \mu \sum_i a_i$. Since the size of T_j is at most $C_2 s$, for every $T_j \in A$, there is at most one C^i s.t. $\delta(C^i, T_j) \leq s$. Clearly, each $T_j \in A$ intersects at least one C^i, thus $\sum_i a_i = |A|$. The proof is complete by combing this with $|\Gamma(A)| \geq \mu \sum_i a_i$.

Now we are ready to prove Theorem 7.

Proof of Theorem 7. Fix constants C_1, C_2. Since $s \geq s_{min}$, by lemma 10, there exists a perfect Δ-matching in H, where $\Delta = \frac{\alpha_{st}(C_2+3)}{C_1}$. Since (i) each red node is assigned with at most Δ blocks, (ii) each block is no larger than $C_2 s$, and (iii) the distance between a block and the red node it is assigned to is at most s, it follows that the total distance we travel in each unit length interval is at most $O(\Delta)s = O(1)s$. Since the the blocks are edge-disjoint subtrees of the MST, the total distance travelled is at most $O(1)\lambda^*(s)$.

4 Conclusion and Future Work

In this paper we presented new results for time window TSP and prize-collecting problems. There are still rooms to improve or obtain good approximation ratios in the general case, or present new inapproximation results for these problems. We hope that these results will be useful in guiding scheduling of autonomous vehicles in practice.

References

1. E. M. Arkin, J. S. Mitchell, and G. Narasimhan. Resource-constrained geometric network optimization. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 307–316. ACM, 1998.
2. S. Arora. Polynomial time approximation schemes for euclidean TSP and other geometric problems. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 2–11. IEEE, 1996.
3. Y. Azar and A. Vardi. TSP with time windows and service time. *arXiv preprint arXiv:1501.06158*, 2015.
4. N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 166–174, 2004.
5. R. Bar-Yehuda, G. Even, and S. M. Shahar. On approximating a geometric prize-collecting traveling salesman problem with time windows. *Journal of Algorithms*, 55(1):76–92, 2005.
6. A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. *SIAM Journal on Computing*, 37(2):653–670, 2007.
7. H.-J. Bockenhauer, J. Hromkovic, J. Kneis, and J. Kupke. The parameterized approximability of TSP with deadlines. *Theory of Computing Systems*, 41(3):431–444, 2007.
8. C. Chekuri, N. Korula, and M. Pál. Improved algorithms for orienteering and related problems. *ACM Transactions on Algorithms (TALG)*, 8(3):23, 2012.
9. K. Chen and S. Har-Peled. The orienteering problem in the plane revisited. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 247–254. ACM, 2006.
10. M. Desrochers, J. K. Lenstra, M. W. Savelsbergh, and F. Soumis. Vehicle routing with time windows: optimization and approximation. *Vehicle routing: Methods and studies*, 16:65–84, 1988.
11. G. Gutin and A. P. Punnen. *The Traveling Salesman Problem and Its Variations*. Springer, 2007.
12. J. S. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999.
13. J. S. B. Mitchell. *Encyclopedia of Algorithms*, chapter Approximation Schemes for Geometric Network Optimization Problems, pages 1–6. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.
14. V. Nagarajan and R. Ravi. Approximation algorithms for distance constrained vehicle routing problems. *Networks*, 59(2):209–214, 2012.
15. J. N. Tsitsiklis. Special cases of traveling salesman and repairman problems with time windows. *Networks*, 22(3):263–282, 1992.

5 Appendix

5.1 Experiments

We have implemented our algorithm for the TWTSP in Euclidean spaces, assuming dyadic time windows. We report briefly on some selected experiments.

First, we tested the running time of the algorithm on random instances. We generated n sites with integer coordinates uniformly distributed in $[0, 1000] \times [0, 1000]$. We assigned each site to have a time window that is randomly chosen from dyadic intervals based on maximum time interval length $L_{max} \in \{2^2, 2^3, \ldots, 2^7\}$. The running times are shown in Table 1. We see that, for fixed L_{max}, the running times (in seconds) grow roughly linearly in n, for $n \in \{2^{14}, 2^{15}, \ldots, 2^{20}\}$. This is consistent with our theoretical time bound of $O(nL_{max} \log n + n^{1.5} L_{max})$.

shown below.

Theorem 8. *The running time of our algorithm for dyadic instance in the plane is* $O(nL_{max} \log n + n^{1.5} L_{max})$.

Proof. For a point set in the plane, an MST can be found in $O(n \log n)$ time by using Delaunay triangulation, hence the time for finding MST on all $O(L_{max})$ groups is $O(nL_{max} \log n)$. Note that the fastest algorithm for maximum bipartite matching is $O(V^{0.5}E)$, where V, E are the number of nodes and edges. Since we have $O(n)$ nodes and $O(nL_{max})$ edges in the bipartite graph we constructed, the time for finding a maximum matching is $O(n^{1.5} \cdot L_{max})$, and the proof is complete.

Since the $O(V^{0.5}E)$-time algorithm bipartite matching algorithm is very complicated, in our implementation, however, we find maximum bipartite matching by reformulating it to a max-flow problem.

Next, we ran tests to compare how well our algorithm performs in terms of its approximation ratio. We compared to two heuristics:

(1) For each site v_i, with dyadic time window $[r_i, d_i]$, we assign v_i to a random integer $X_i \in [r_i, d_i - 1]$. This results in each site v_i being assigned to a cluster, associated with the integer X_i. We then compute for each cluster $j \in [1, L_{max} - 1]$ an approximate TSP path through the sites associated with cluster j, and concatenate these paths in order, by j.

(2) For each site v_i, with dyadic time window $[r_i, d_i]$, we assign v_i to the integer $j \in [r_i, d_i - 1]$ that minimizes the distance between v_i and the set of sites whose time window equals $[j, j+1]$. This results in each site v_i being assigned to a cluster, associated with an interval $[j, j + 1]$. We then compute for each cluster $j \in [1, L_{max} - 1]$ an approximate TSP path through the sites associated with cluster j, and concatenate these paths in order, by j.

In both of these two algorithms, we used the well known factor-2 MST-traversal algorithm for finding approximate TSP path. Table 2 shows the result of running

our algorithm and the two heuristics on an instance (from the TSP Library) whose sites are a set of 4463 cities in Canada. For this experiment, we set $L_{max} = 7$. We show the total length of the path computed, the ratio of the length to the optimal length ($\lambda^*(s) = 1290.032$, which was given in the library), and the maximum speed utilized by the solutions.

We see that our algorithm yielded a shorter path than did Heuristic (1), using only slightly higher speed. While Heuristic (2) achieved a shorter length than our algorithm, it did so at a cost of using a much greater speed; this heuristic does poorly at controlling the speed, since it does not assign points to clusters in a "balanced" manner, as does our algorithm; in fact, it can end up having arbitrarily large speeds.

Table 1. Running Time Test (in seconds)

		$\log_2 L_{max}$					
		2	3	4	5	6	7
$\log_2 n$	14	1.585	1.659	2.038	2.561	3.667	5.960
	15	3.226	3.503	4.098	5.183	7.113	11.149
	16	6.859	8.010	8.711	10.912	14.798	22.850
	17	14.298	15.853	18.036	22.018	30.136	49.199
	18	36.903	39.510	46.926	59.194	84.631	125.104
	19	61.954	67.883	78.914	99.073	135.869	276.448
	20	124.860	144.1080	169.673	216.227	312.26	968.360

Table 2. Approximation Ratio and Maximum Speed ($\lambda^*(s) = 1290.032$)

		length	ratio with $d^*(s)$	max speed
Algorithm	Our Algorithm	7472.677	5.791	273.322
	Heuristic 1	17959.672	13.922	211.029
	Heuristic 2	4060.160	3.147	724.704

5.2 Proof of Lemma 3

Lemma 3 (Polynomial Boundary Complexity of P^*) Let $P^* = OPT(\mathcal{S}([a,b]); \theta)$, $B = B(P^*, a, b)$, $B_L = B(P^*, a, c)$ and $B_R = B(P^*, c, b)$. Let τ_b, τ_e be defined as in Definition 2. Then, P^* leaves/enters each of $\partial^N B_L$, $\partial^S B_L$, $\partial^N B_R$, $\partial^S B_R$ at most once in $[\tau_b, \tau_e]$.

Proof. We only prove the following statement, since other parts are similar: P^* leaves/enters $\partial^N B_L$ at most once in τ_b, τ_e. (Recall that $\tau_b = \min\{\tau \in [a,b]: P(\tau) \in [y_S, y_N]\}$, and $\tau_e = \max\{\tau \in [a,b]: P(\tau) \in [y_S, y_N]\}$.) Let $I = [a,b]$. Suppose a path P leaves B_L more than once in $\tau_{b,L}, \tau_{e,L}$ (see the blue path in

Figure 5). Let Q_i be the components of P B_L, i.e. portions P outside B_L. Let Q_k be the highest portion. By definition of B_L, each Q_i visits no segment in $\mathcal{S}(I_L)$. Since any segment that Q_i visits in $V[a,b]$ must also be visited by Q_k, we can remove Q_i and touch no less segments.

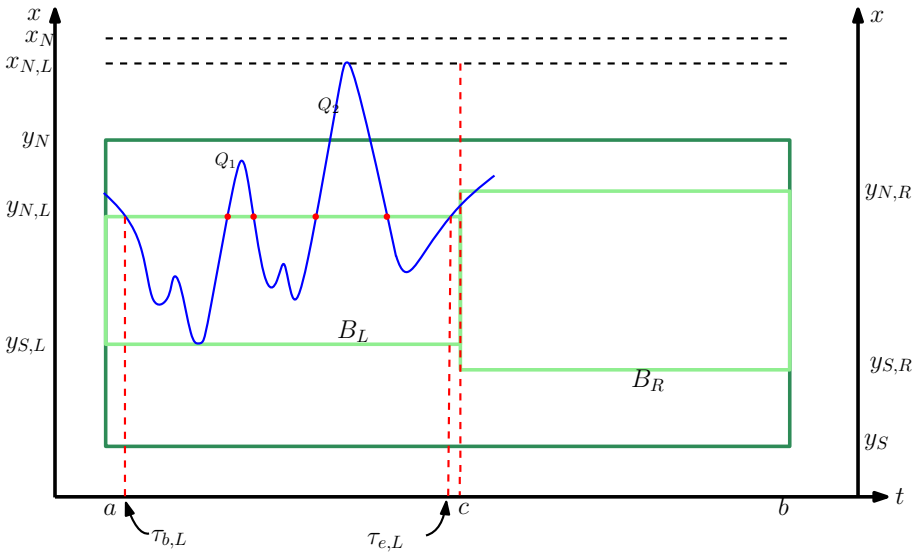

Fig. 5. Illustration of the "polynomial boundary complexity" lemma. The two light green boxes are $B_L = [a,c] \times [y_{S,L}, y_{N,L}]$ and $B_R = [c,b] \times [y_{S,R}, y_{N,R}]$, while the dark green one is B. Let $P^* = OPT(I;\theta)$. Suppose between $[\tau_b, \tau_e]$, the left path $P^*|_L$ has 2 pieces outside B_L, denoted by Q_1, Q_2. In this case, we can remove Q_1 without losing any prize in $\mathcal{S}(I)$. Indeed, one the one hand, by definition of B_L, Q_i's do not touch any segment in $\mathcal{S}(I_L)$, so flattening Q_1 will not reduce the number of segments P^* touches in $\mathcal{S}(I_L)$. On the other hand, since any segment in $V[a,b]$ that Q_1 touches is also touched by Q_2, so flattening Q_1 does not reduce the number of segments in $V[a,b]$ that P^*.

5.3 Proof of Lemma 4

Lemma 4. The 1D TWPC problem with dyadic time windows can be solved in $poly(n, L_{max})$ time.

Proof. Let us start by showing a simple example where the dynamic programming for TWTSP no longer holds for TWPC. Consider the 1D TWTSP problem with $s = \infty$ for a dyadic instance, we will use the notation $OPT(I;\theta)$ as in the last section for now. Recall that if $P^* = OPT(I;\theta)$ for some dyadic interval I, where θ encodes the highest/lowest positions visited by P, then

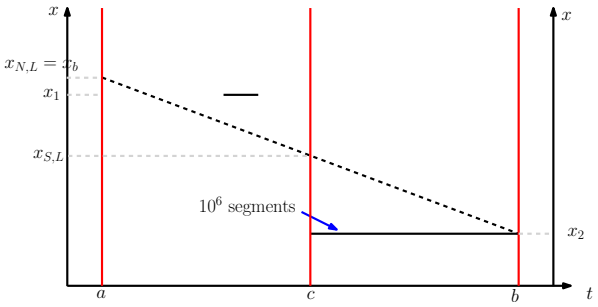

Fig. 6. a simple example where the dynamic programming for TWTSP no longer holds for TWPC. The dashed slanted line is the optimal solution.

$P^*|_{I_L} = OPT(I; \theta_L^*)$ where θ_L^* is the parameter induced by $P^*|_{I_L}$. But for the finite speed TWPC problem this is not the case.

Consider the example in Fig 6. Suppose there are a million and one segments in our instance. Suppose $|x_b - x_2|$ is such that if we move towards x_2 immediately, we can touch those segments just in time. Thus, any optimum path P^* should ignore the segment at location x_1, otherwise it will not have enough time to travel to x_2. Note that $P^*|_{I_L}$ is **not** $OPT(I_L; \theta_L^*)$, where θ_L^* is the parameter it induces on I_L, since $P^*|_{I_L}$ collects 0 prize while $OPT(I; \theta_L^*)$ is 1.

The proof is similar to the one for TWTSP. Given a dyadic interval $I = [a, b]$, we define $OPT(I, \theta)$ as in Definition 3. Given $\theta, \theta_L, \theta_R$, where

$$\theta = (y_N, y_S; x_N, x_S; x_b, x_e; \tau_b, \tau_e, \tau_N^-, \tau_N^+, \tau_S^-, \tau_S^+),$$
$$\theta_L = (y_{N,L}, y_{S,L}; x_{N,L}, x_{S,L}; x_{b,L}, x_{e,L}; \tau_{b,L}, \tau_{e,L}, \tau_{N,L}^-, \tau_{N,L}^+, \tau_{S,L}^-, \tau_{S,L}^+),$$
$$\theta_R = (y_{N,R}, y_{S,R}; x_{N,R}, x_{S,R}; x_{b,R}, x_{e,R}; \tau_{b,R}, \tau_{e,R}, \tau_{N,R}^-, \tau_{N,R}^+, \tau_{S,R}^-, \tau_{S,R}^+),$$

we say θ_L, θ_R are *compatible* with θ, if there exists a path P with parameter θ, such that $P|_{I_L}$ and $P|_{I_R}$ have parameter θ_L and θ_R respectively. Clearly, given $\theta_L, \theta_R, \theta$, it takes constant time to check their compatibility. Define

$$g(a, b; x_N, x_S) = \#\{s \in V[a, b] : x(s) \in [x_S, x_N]\}.$$

We claim that $P^*|_{I_L} = OPT(I_L; \theta_L)$, where θ_L is the parameter induced by P_L. The similar also holds for $P^*|_{I_R}$.

We prove the claim above by the following swapping argument. Given a path P, let $A_L(P)$, $A_R(P)$, $A_{mid}(P)$ be the number of segments it visits in $\mathcal{S}(I_L)$ and $\mathcal{S}(I_R)$, $V[a, b]$ respectively. Then, the number of segments in $\mathcal{S}(I)$ that it visits equals $A_L(P) + A_R(P) + A_{mid}(P)$. So if $P' := OPT(I_L; \pi(I_L); \theta_L)$ visits more segments in $\mathcal{S}(I_L)$ than $P^*|_{I_L}$, then $P' \cup P^*|_{I_R}$ visits more segments in $\mathcal{S}(I)$ than P^*. Note that $A_{mid}(P), A_L(P)$ and $A_R(P)$ are completely determined by θ_L and θ_R, and $P^*|_{I_L}, P'$ share the same parameter θ_L, so $P' \cup P^*|_{I_R}$ is also feasible, contradiction. Hence, we have the following recursive relation:

$$OPT(I; \theta) = \max\{OPT(I_L; \theta_L) + OPT(I_R; \theta_R) + g(a, b; x_N, x_S) : \theta_L, \theta_R \textit{ compatible with } \theta\}.$$

Clearly there are $O(n^{24})$ combinations of parameters (θ_L, θ_R), and for each combination, it takes polynomial time to check whether it is compatible with θ

and compute $g(a, b; x_N, x_S)$. Clearly, there are $O(n^{12})$ different parameters for each dyadic I, and $O(L_{max})$ dyadic intervals in each row in the lookup table. Since there are $O(\log L_{max})$ rows in the lookup table, the proof is complete.

5.4 Proof of Theorem 2

Theorem 2 Given an instance for 1D TWPC problem with bounded velocity s, let L_{max} be the maximum length of the input segments, and assume the shortest time window has length ≥ 1. Then for any $\varepsilon > 0$, in $O((nL_{max})^{O(\frac{\log L_{max}}{\log(1+\varepsilon)})})$ time we can find a path P, such that

1. the number of segments that P visits is at least k^*,
2. each segment s_i is visited in $[r_i - \varepsilon L_i, d_i + \varepsilon L_i]$, where $L_i = d_i - r_i$.

Definition 3. *Given a dyadic interval $I = [a, b]$, and an h-partition $\pi = \{t_i\}$ on it. Also given the following sequences, each of length h:*

$$X_N = \{x_N^i\}_{1 \leq i \leq h},$$

$$X_S = \{x_S^i\}_{1 \leq i \leq h},$$

$$X_b = \{x_b^i\}_{1 \leq i \leq h},$$

$$X_e = \{x_e^i\}_{1 \leq i \leq h},$$

$$T_N^- = \{\tau_N^{-,i}\}_{1 \leq i \leq h},$$

$$T_N^+ = \{\tau_N^{+,i}\}_{1 \leq i \leq h},$$

$$T_S^- = \{\tau_S^{-,i}\}_{1 \leq i \leq h},$$

$$T_S^+ = \{\tau_S^{+,i}\}_{1 \leq i \leq h},$$

$$T_b = \{\tau_b^i\}_{1 \leq i \leq h},$$

$$T_e = \{\tau_e^i\}_{1 \leq i \leq h}.$$

Define $OPT(I; \theta)$ where $\theta = (y_N, y_S; X_b, X_e; T_N^-, T_N^+, T_S^-, T_S^+; T_b, T_e)$ as the optimal solution to the following problem:

Maximize number of segments in $\mathcal{S}(I)$ visited by P, s.t.

1. *the max/minimum x-coordinates that P visits in $[t_{i-1}, t_i]$ are x_N^i and x_S^i respectively,*
2. *the x-coordinates of the highest/lowest segment in $\mathcal{S}(I)$ that P visits are y_N and y_S respectively,*
3. *the initial and ending position in $[t_{i-1}, t_i]$ of P are x_b^i and x_e^i respectively,*
4. *$\tau_b^i = min\ \{\tau \in [t_{i-1}, t_i]$: $P(\tau) \in [y_S, y_N]\}$, $\tau_e^i = max\ \{\tau \in [t_{i-1}, t_i]$: $P(\tau) \in [y_S, y_N]\}$,*

5. *In each* $[\tau_b^i, \tau_e^i]$, P *arrives at* y_N *at time* $\tau_N^{-,i}$ *and moves to* x_N^i, *and comes back to* y_N *at time* τ_N^i, *it also arrives at* y_S *at time* $\tau_S^{-,i}$ *and moves to* x_S^i, *and comes back to* y_S *at time* $\tau_S^{+,i}$, *and finally arrives at* x_e^i.

As a subroutine, we first prove the following result.

Theorem 9. *The 1D TWPC problem with h-dyadic time windows can be solved in* $O((nL_{max})^{O(h)})$ *time.*

Similar to the dyadic case, the proof relies on the fact that any reasonable path has small boundary complexity. The following is a straightforward generalization of **Lemma 3**.

Lemma 11. *(Boundary Complexity of* P^**) Let* $P^* = OPT(\mathcal{S}([a,b]); \theta)$, $B = B(P^*, a, b)$, $B_L = B(P^*, a, c)$ *and* $B_R = B(P^*, c, b)$. *Let* T_b, T_e *be defined as in Definition 3. Then for each* $i \le h$, P^* *leaves/enters each of* $\partial^N B_L$, $\partial^S B_L$, $\partial^N B_R$, $\partial^S B_R$ *at most once between* τ_b^i, τ_e^i.

Proof of Theorem 9. Mimic the proof for dyadic case. Let $I = [a,b]$ be a dyadic interval with midpoint c. Define θ as in Definition 3. Given $\theta, \theta_L, \theta_R$, where

$$\theta = (y_N, y_S; X_b, X_e; T_N^-, T_N^+, T_S^-, T_S^+; T_b, T_e),$$
$$\theta_L = (y_{N,L}, y_{S,L}; X_{b,L}, X_{e,L}; T_{N,L}^-, T_{N,L}^+, T_{S,L}^-, T_{S,L}^+; T_{b,L}, T_{e,L}),$$
$$\theta_R = (y_{N,R}, y_{S,R}; X_{b,R}, X_{e,R}; T_{N,R}^-, T_{N,R}^+, T_{S,R}^-, T_{S,R}^+; T_{b,R}, T_{e,R}),$$

we say θ_L, θ_R are *compatible* with θ, if there exists a path P with parameter θ, such that $P|_{I_L}$ and $P|_{I_R}$ have parameter θ_L and θ_R respectively.

The proof follows immediately from the following recursive relation, which can be easily verified with a swapping argument similar in the dyadic case:

$$OPT(I; \pi; \theta) = \max_{\theta_L, \theta_R} \{OPT(I_L; \pi_L; \theta_L) + OPT(I_R; \pi_R; \theta_R) + \sum_{1 \le i \le c < j \le h} g_{ij}\},$$

where

$$g_{ij} := g(t_i^L, t_j^R; \max_{i \le l \le j-1} \{x_N^l\}, \min_{i \le l \le j-1} \{x_S^l\}),$$

and θ_L, θ_R are compatible with θ.

5.5 Proof of Theorem 2

Theorem 2. Given an instance for 1D TWPC problem, let L_{max} be the maximum length of the input segments, and assume the shortest time window has length ≥ 1. Then for any $\varepsilon > 0$, in $O((nL_{max})^{O(\frac{\log L_{max}}{\log(1+\varepsilon)})})$ time we can find a path P, such that

1. the number of segments that P visits is at least k^*,
2. each segment s_i is visited in $[r_i - \varepsilon L_i, d_i + \varepsilon L_i]$, where $L_i = d_i - r_i$.

Proof. Very similar to TWTSP. First construct a family of ε-dense partition Π. Then, transform our instance into an h-dyadic instance $\mathcal{I}'$, where $h = O(\frac{\log L_{max}}{\log(1+\varepsilon)})$, by stretching each segment s_i so that both its endpoints are the partition points of $\pi(W(s_i))$. Find an optimal solution P for $\mathcal{I}'$ in $O(n^{O(h)})$ time. Clearly $|P| \geq \lambda$. Note that we have stretched each segment by at most $1+\varepsilon$ times, so P visits each segment s_i is visited in $[r_i - \varepsilon L_i, d_i + \varepsilon L_i]$.

5.6 Proof of Lemma 2

Lemma 2. Let $L_{max} > 0$ be a power of 2. Then we can assign a partition $\pi(I)$ to each dyadic interval $I \subseteq [0, L_{max}]$, such that

1. $\Pi = \{\pi(I)$: dyadic $I \subseteq [0, L_{max}]\}$ is a family of partitions with the inheriting property,
2. each $\pi(I)$ has at most $O(\frac{\log L_{max}}{\log(1+\varepsilon)})$ partition points,
3. $\pi(I)$ is ε-dense for each dyadic $I \subseteq [0, L_{max}]$.

Proof. See Fig 7 and Fig 8. Recall that we say a dyadic interval I is on level l, if its length is 2^l. Let $l_{max} := O(\log L_{max})$. We will construct this family of partitions Π "dynamically": at iteration $l = 0, 1, ..., l_{max}$, we construct $\pi(J)$ for every dyadic interval J on level l, and in order to maintain the inheriting property, we also add some new partition points to $\pi(I)$ for every offspring I of J.

Let $\pi^l(I)$ be the partition associated with I after l iterations in our construction process, and $H_l(I)$ be the number of partition points in $\pi^l(I)$. Note that if $l <$ level(I) then $\pi^l(I) = \emptyset$, because we have not yet started constructing $\pi(I)$. Also note that for any fixed I, $H_l(I)$ is a non-decreasing with respect to l.

We construct Π as follows: at iteration l, for an interval J on level l,

(1) copy the left half of $\pi^l(J_L)$ to the leftmost quarter of J, and similarly the right half of $\pi^l(J_R)$, to the rightmost quarter of J. (see Figure 7.)

(2) add $O(\frac{1}{\log(1+\varepsilon)})$ extra partition points to J in the middle, in order to ensure the $\varepsilon-$dense property. In fact, let L be the length of I. Since $\log_{1+\varepsilon} L$ - $\log_{1+\varepsilon} \frac{L}{2} = \frac{\log 2}{\log(1+\varepsilon)}$, we know that the number of partition points added in the middle is $\frac{\log 2}{\log(1+\varepsilon)} = O(\frac{1}{\log(1+\varepsilon)})$. (see Figure 7)

(3) In order to maintain the inheriting property, we also add extra $O(\frac{1}{\log(1+\varepsilon)})$ partition points to all offsprings of J, and maintain the symmetry of each partition by adding $O(\frac{1}{\log(1+\varepsilon)})$ more partition points to each offspring, see Figure 8.

From the description of the construction, we know that for J on level l,

$$H_l(J) \leq \frac{1}{2}H_{l-1}(J_L) + \frac{1}{2}H_{l-1}(J_R) + O(\frac{1}{\log(1+\varepsilon)}),$$

and for any I and $l \geq$ level(I),

$$H_{l+1}(I) \leq H_l(I) + O(\frac{1}{\log(1+\varepsilon)}).$$

Then, for any J with $\text{level}(J) = l \leq l_{max}$, we have

$$H_{l_{max}}(J) \leq H_l(J) + O(\frac{l_{max} - l}{\log(1+\varepsilon)}).$$

In particular, when $l = 0$,

$$H_{l_{max}}(J) \leq O(1) + O(\frac{l_{max}}{\log(1+\varepsilon)}) = O(\frac{\log L_{max}}{\log(1+\varepsilon)}).$$

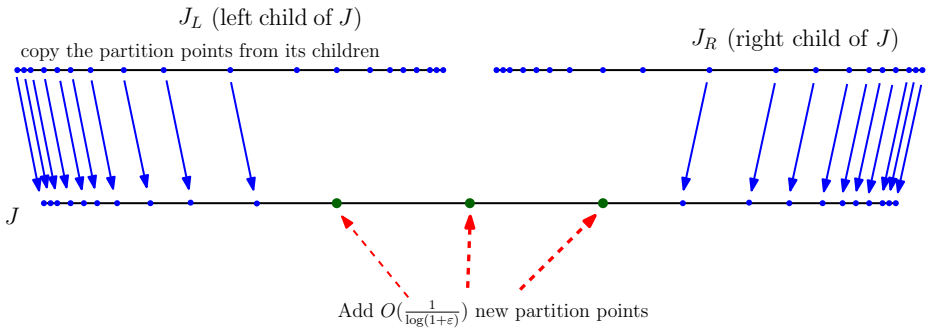

Fig. 7. Illustration of the dynamic construction of ε-dense partitions, stage 1. The red partition points in J are directly copied from J_L and J_R. Then we add the green points to J in order to make $\pi(J)$ ε-dense.

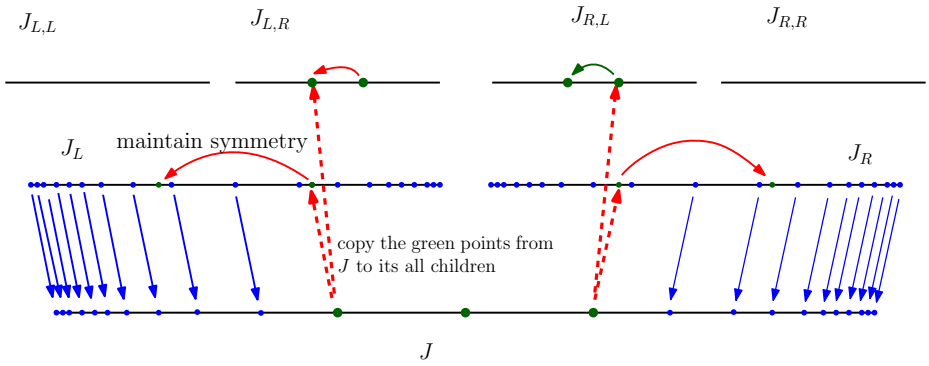

Fig. 8. Illustration of stage 3. in order to maintain the inheriting property, add the green point back to all offsprings of J.

5.7 Proof of Theorem 6

Lemma 12. *If all time windows are unit length, then there is constant factor approximation for the TWTSP problem in metric space.*

Proof. Consider the following algorithm: for integer t, let S_t be the points whose time window contains t. Pick any two points a_t, b_t in S_t for each t, and find an $O(1)$-approximation P_t for the point-to-point TSP problem for S_t, starting at a_t and ending in b_t. Then concatenate the paths P_t's by connecting b_t with a_{t+1}. Then the total distance is $\sum_t \delta(b_t, a_{t+1}) + \sum_t |P_t|$.

We claim that this value is most $O(1) \cdot \lambda^*$. Let P^* be an optimal solution, and $P^*_{[t,t']}$ be its restriction on time window $[t, t']$. Clearly, $|P_t| \leq O(1) \cdot |P^*_{[t-1,t+1]}|$ for each t, since $P^*_{[t-1,t+1]}$ must visit all points in S_t. On the other hand, $\delta(b_t, a_{t+1}) \leq |P^*_{[t-1,t+2]}|$ since both b_t and a_{t+1} are visited by P^* in $[t-1, t+2]$. Therefore, $\sum_t \delta(b_t, a_{t+1}) + \sum_t |P_t| \leq \sum_t O(1) \cdot |P^*_{[t-1,t+1]}| + \sum |P^*_{[t-1,t+2]}| \leq O(1)|P^*| = O(1) \cdot \lambda$.

5.8 Proof of Lemma 7

Lemma 7. In a bipartite graph $G = (V_R \cup V_B, E)$, if for any subset X of V_B, the neighboring vertices set $\Gamma_G(X)$ satisfies $|\Gamma_G(X)| \geq \frac{1}{\Delta}|X|$, then there exists a perfect Δ-matching.

Proof. Construct an auxiliary graph G' as follows: create Δ copies of each node in V_R and add dummy edges correspondingly[1]. Clearly, for any subset X of V_B, $|\Gamma_{G'}(X)| = \Delta \cdot |\Gamma_G(X)|$. Hence, $|\Gamma_{G'}(X)| \geq \Delta \cdot \frac{1}{\Delta} \cdot |\Gamma_G(X)| = |\Gamma_G(X)|$. By Hall's marriage theorem, there exists a perfect matching M for V_B in G'. Mapping M back to a G by contracting all edges incident to the same red supernode in G', we obtain the desired perfect Δ-matching in G.

5.9 Proof of Lemma 8

Lemma 8. Given a weighted undirected graph $G = (V, E)$, let $T_1, ...T_m$ be some node-disjoint subtrees of $MST(G)$. Let $V' = \cup_{1 \leq i \leq m} V(T_i)$ and G' be the restriction of G on V'. Then $MST(G') \geq \sum_i |T_i|$.

Proof. Without loss of generality, assume there is no tie in the edge lengths. We will show that for any edge e in some T_i, e is also contained in $MST(V')$. Argue by contradiction. Suppose when we encounter e in Kruskal's algorithm for G', we find that adding it to the cuurent forest will create a cycle. Then, there must be a cycle C in $F \cup e$, where F is our current forest. Since G' is a subgraph of G, we know that C is also a cycle of G. Therefore, we have found a cycle in G whose largest edge is e. Clearly, this implies that e is not in $MST(G)$. But e is in some T_i, and hence in $MST(G)$. Contradiction.

[1] i.e. if (r_i, b_j) is an edge in G, then add edges (r_i^k, b_j) to G' for $k = 1, 2, ...\Delta$, where r_i^k are the copies of r_i.

5.10 Proof of Theorem 5

Theorem 5. Under the assumption above, if $s \geq s_{min}$, then there is an $(O(\log L_{max}), O(\log L_{max}))$ dual approximation for the TWTSP problem for dyadic instance.

Proof. Let V^j be set of all vertices with time window length 2^j. For each $j \leq \log L_{max}$, consider $V^1 \cup V^j$, assign constant number of blocks to Red_i's using algorithm in the last subsection. Then, each Red_i is assigned with $O(\log L_{max})$ blocks in the end. So we can use $O(\log L)$ speed to traverse the blocks assigned to each Red_i. On the other hand, the distance travelled is $O(1)\sum_j MST(V^1 \cup V^j) = O(\log L)\lambda^*(s)$.

5.11 Figures

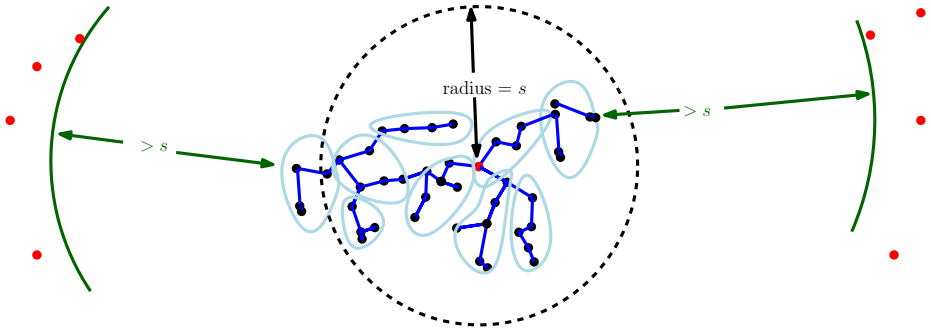

Fig. 9. The intuition of lemma 10. Consider this toy example: for each block in the figure, there is only one red node r within distance s from it. Then, r is "responsible" for all the black nodes in these blocks, hence has to travel a large distance. But its black cycle can only travel at most s. In other words, it does not have enough "energy" to fulfill its responsibility. So there should not be a feasible solution!

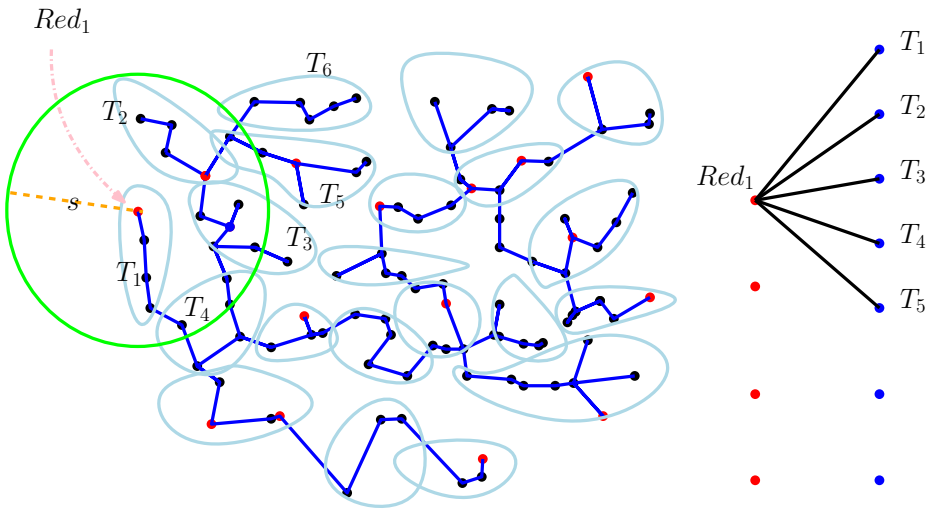

Fig. 10. Illustration of the construction of H. The green circle centered at Red_1 has radius s. For any blue block intersecting this green circle, i.e. within distance s from it, we add an edge between this block with the red node r_1.