数理解析研究所講究録
1489巻 2006年 174-180

Single Vehicle Scheduling Problem for Processing a Maximum Benefit Subset of Jobs on a Line

Takaharu Ohnishi (大西隆治), Hiroshi Nagamochi (永持仁)
Department of Applied Mathematics and Physics
Graduate School of Informatics, Kyoto University
Sakyo, Kyoto 606-8501, Japan

Abstract

In this paper, we study a problem of finding a scheduling for a single vehicle to process jobs located on a single line, where each job has a handling time, a time window (a pair of a release time and a deadline), and a benefit. The objective is to maximize the total benefit from jobs processed by the vehicle. We present an $O(\gamma n^{(3+\rho)})$ time 2-approximation algorithm for the scheduling problem, where n is the number of jobs, γ is the ratio of the maximum length of a time window to the minimum handling time, and ρ is the maximum number of jobs that can be processed during the time period after processing a job j and before visiting the job j again by the deadline of j.

1 Introduction

Vehicle Routing Scheduling Problem (VSP) has been studied as one of the most important scheduling problems [3, 4]. We are given a set J of n jobs (such as items to be picked up or facilities to be inspected). Each job is characterized by a *release time*, a *handling time*, a *deadline* and a *benefit*. For a job, the time interval between its release time and deadline may be called its *time window*. A handling time means the time required for processing its job, where no interruption is allowed during a process of any job. The problem asks to find a schedule which minimizes (or maximizes) an objective function such as the completion time of processing all jobs, the maximum lateness from deadlines, and so on. The *tour version* of the VSP requires each vehicle to return to its initial position at the end of a schedule while *the path version* allows each vehicle to stay at any position at the end of a schedule.

We consider a single vehicle scheduling problem such that all jobs are located on a single line and each job has a handling time, a time window, and a benefit. We call this problem VSP-PATH (resp., MAX-VSP-PATH) if the objective is to minimize the makespan (resp., to maximize the total benefit from jobs processed by the vehicle). These problems have important applications such as ship scheduling, where a ship picks up cargoes along a shoreline [9], and truck scheduling, where a truck delivers goods to customers along a highway [6]. Cargoes along a shoreline or customers along a highway can be regarded as jobs on a straight line. Tables 1 and 2 show a summary of the known results on several special cases of the VSP-PATH and the MAX-VSP-PATH, respectively.

Table 1: Results on special cases of the VSP-PATH.

handling time $h(j)$	time window ($r(j)$: release time, $d(j)$: deadline)		
	arbitrary $r(j) \le d(j)$	$r(j) \equiv 0$	$d(j) \equiv \infty$
arbitrary $h(j)$	strongly NP-hard (path version) [5]	open	NP-hard [10] (path version)
$h(j) \equiv 0$	strongly NP-hard (path version) [10]	$O(n^2)$ [10] (path version)	$O(n^2)$ [9] (path version) $O(n)$ [9] (tour version)

In this paper, we consider the path version of the MAX-VSP-PATH with general time windows, *nonzero* handling times, and general benefits. We denote by γ the ratio of the maximum length of a time window to the minimum handling time and by ρ the maximum number of jobs that can be processed during the time period after processing

Table 2: Results on special cases of the MAX-VSP-PATH.

handling time $h(j)$		time window ($r(j)$: release time, $d(j)$: deadline)	
		arbitrary $r(j) \leq d(j)$	$r(j) \equiv d(j)$
	$h(j) > 0$	strongly NP-hard weakly NP-hard ($\rho = 0$)	open
		2-approximable in $O(\gamma n^{(3+\rho)})$ time in $O(\gamma n^2)$ time ($\rho = 0$)	
	arbitrary $h(j)$	strongly NP-hard [5]	open
	$h(j) \equiv 0$	strongly NP-hard [10]	$O(n \log n)$[1]
		$O(\log L)$-approximable [2]	
	$h(j) \equiv 1$	8-approximable ($d(j) - r(j)$: uniform) [2]	(unit benefit)

a job j and before visiting the job j again by the deadline of j. The results obtained in this paper are as follows. We first prove that the MAX-VSP-PATH with nonzero handling times is weakly NP-complete even if no two jobs j and j' are located at close positions on the line in the sense that j cannot be visited by the vehicle by its deadline once j and j' are processed in this order. Such an instance is called *sparse*, where ρ of a sparse instance is 0. Next, we present an $O(\gamma n^{(3+\rho)})$ time 2-approximation algorithm. Moreover, we present an $O(\gamma n^2)$ time 2-approximation algorithm for sparse instances. To derive these results, we first convert the MAX-VSP-PATH into a problem of finding a monotone curve in the x, y-plane, where a time window and a schedule for a vehicle are represented as a line segment of gradient 1 and a monotone curve, respectively. Processing a job j is represented as a curve in the x, y-plane that intersects the segment of the time window of j. To find a monotone curve that intersects a maximum benefit set of line segments, we construct a digraph, called a *chart graph*, such that each directed path in the digraph is a monotone curve. We prove that there exists a digraph with size $O(\gamma n^2)$ that contains a 2-approximate solution.

The remainder of this paper is organized as follows. Section 2 introduces the MAX-VSP-PATH and reduces this to a problem of finding a monotone curve that intersects a maximum benefit set of segments in the x, y-plane. Section 3 defines a chart graph, in order to find an approximate solution to the problem of finding an optimal monotone curve. Section 4 proves that there exists a chart graph that has a 2-approximate monotone curve. Section 5 proposes an approximation algorithm by using dynamic programming. Section 6 makes some concluding remarks.

2 Problem Description

In this section, we first formulate the MAX-VSP-PATH, a problem of finding a single vehicle scheduling to process jobs so as to maximize the total benefit from the processed jobs. Let $\mathbb{R}$ and $\mathbb{R}_+$ be the sets of reals and nonnegative reals, respectively. An instance $I = (J, r, d, h, b, l)$ of the MAX-VSP-PATH consists of a set J of n jobs, a *release time* $r(j) \in \mathbb{R}_+$, a *deadline* $d(j) \in \mathbb{R}_+$, a *handling time* $h(j) \in \mathbb{R}_+$, a *benefit* $b(j) \in \mathbb{R}_+$, and a *position* $l(j) \in \mathbb{R}$ for each job $j \in J$, where time interval $[r(j), d(j)]$ is called the *time window* of job $j \in J$.

We are given a single vehicle which is initially situated at position 0 and time 0. A job $j \in J$ can be processed if it is visited by the vehicle during the time window $[r(j), d(j)]$, and cannot be processed otherwise. It takes $h(j)$ time to finish a process of a job j. The *travel time* for the vehicle to travel from a job $j \in J$ to a job $j' \in J$ is by $|l(j) - l(j')|$. Moreover, the vehicle gets benefit $b(j)$ if it processes job j. A schedule σ for a subset $J' \subseteq J$ of jobs is a bijection $\sigma : \{1, \ldots, |J'|\} \to J'$, and is called *feasible* if all jobs can be processed by a single vehicle which starts from position 0 at time 0 and visits J' in the order of $\sigma(1), \sigma(2), \ldots, \sigma(|J'|)$. In other words, a schedule σ for $J' \subseteq J$ is feasible if it admits arrival times $t(\sigma(i))$ at jobs $\sigma(i)$, $i = 1, \ldots, |J'|$, such that

$$t(\sigma(i)) \in [r(\sigma(i)), d(\sigma(i))], \text{ and}$$
$$t(\sigma(i)) \geq t(\sigma(i-1)) + h(\sigma(i-1)) + |l(\sigma(i-1)) - l(\sigma(i))|,$$

where $t(\sigma(0)) = h(\sigma(0)) = 0$. The objective of this problem is to find a feasible schedule σ for a subset $J' \subseteq J$ with the maximum benefit $\Sigma_{j \in J'} b(j)$.

Throughout the paper, we assume that $J \neq \emptyset$ and

$$h(j) > 0,\ b(j) > 0,\ r(j) \geq |l(j)| \text{ for all } j \in J. \quad (1)$$

Note that the vehicle starting from position 0 at time 0 cannot reach position $l(j)$ earlier than time $|l(j)|$. Let $h_{\min} = \min_{j \in J} h(j)$. Define

$$\gamma = \left\lfloor \frac{\max_{j \in J}(d(j) - r(j))}{h_{\min}} \right\rfloor + 1. \quad (2)$$

Moreover, let ρ be the maximum number of jobs that can be processed during the time period after processing a job j before visiting the job j by

deadline of j. We call an instance I *sparse* if $\rho = 0$, i.e., no two jobs j and j' are located at close positions on the line in the sense that j cannot be visited by the vehicle by its deadline once j and j' are processed in this order.

2.1 NP-hardness of the MAX-VSP-PATH

In this section, we prove the NP-hardness of special cases of the MAX-VSP-PATH.

Theorem 2.1 *The MAX-VSP-PATH is strongly NP-hard even if all jobs have the same length of time windows and the same benefit.*

PROOF: We can show a polynomial reduction from the 3-PARTITION [5] to the problem (the detail is omitted due to space limitation). ∎

Theorem 2.2 *The MAX-VSP-PATH is weakly NP-hard even for sparse instances.*

PROOF: We can show a polynomial reduction from the Knapsack problem [5] to the problem (the detail is omitted due to space limitation). ∎

2.2 Monotone Curves in the Plane

We represent a problem instance I of the MAX-VSP-PATH in the x, y-plane as follows. We consider the x, y-plane $\mathbb{R}^2_+$ with nonnegative coordinates, where the origin of the x, y-plane, denoted by s_0, represents position 0 at time 0. A pair of position l on the line and time t is mapped to point $((t+l)/\sqrt{2}, (t-l)/\sqrt{2})$ in the x, y-plane. For each job $j \in J$, define points

$$a(j) = ((r(j) + l(j))/\sqrt{2}, (r(j) - l(j))/\sqrt{2}),$$

and

$$a'(j) = ((d(j) + l(j))/\sqrt{2}, (d(j) - l(j))/\sqrt{2})$$

and a line segment $\tau(j) = [a(j), a'(j)]$ of gradient 1 in the x, y-plane. A pair of time window $[r(j), d(j)]$ and position $l(j)$ is mapped to $\tau(j)$ in the x, y-plane (see Fig. 1), where we may also denote by $\tau(j)$ the set of points on the line segment. For each $j \in J$, let h_j denote the vector $(h(j)/\sqrt{2}, h(j)/\sqrt{2})$ and let h^* denote the vector $(h_{\min}/\sqrt{2}, h_{\min}/\sqrt{2})$ in the x, y-plane.

For a point $c = (c_x, c_y) \in \mathbb{R}^2_+$, let

$$R(c) = \{c' = (c'_x, c'_y) \mid c_x \le c'_x,\ c_y \le c'_y\},$$

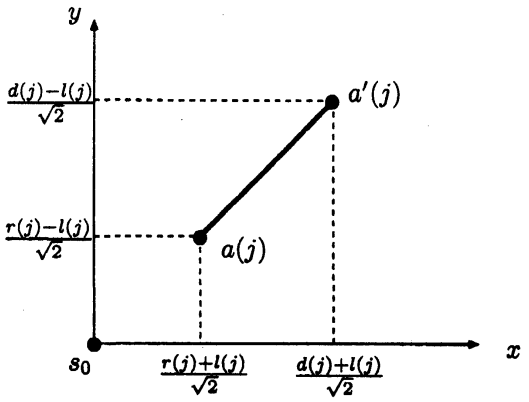

Figure 1: Time window $\tau(j) = [a(j), a'(j)]$ of job j in the x,y-coordinate.

$$L^r(c) = \{(c'_x, c_y) \mid c_x \le c'_x\},$$
$$L^u(c) = \{(c_x, c'_y) \mid c_y \le c'_y\}.$$

Note that $L^r(c)$ (resp., $L^u(c)$) denotes the half line starting from point c in the rightward (resp., upward) direction. We define a transitive relation $\preceq$ by

$$c \preceq c' \Leftrightarrow c' \in R(c) \quad \text{for} \quad c, c' \in \mathbb{R}^2_+.$$

Let $\Pi(s_0)$ be the set of all monotone curves starting from the origin s_0 in the x, y-plane. We say that a curve $\pi \in \Pi(s_0)$ *visits* $\tau(j)$ if π contains a line segment $\bar{\pi}$ of gradient 1 and of length $h(j)$ that intersects $\tau(j)$ (see Fig. 2), where $\bar{\pi}$ is not necessarily contained in $\tau(j)$ completely. Although a monotone curve $\pi \in \Pi(s_0)$ may have more than one such line segment $\bar{\pi}$, we say that π *collects* $\tau(j)$ and *gets* benefit $b(j)$ when it visits $\tau(j)$ for the first time, and denote by $g_\pi(j)$ and $f_\pi(j)$ the beginning point and finishing point of such $\bar{\pi}$, i.e., $\bar{\pi} = [g_\pi(j), f_\pi(j)]$. Note that, for the same j, $\tau(j)$ is not collected more than once.

We easily obtain the following observation.

Lemma 2.3 *Let J' be a subset of jobs. There is a monotone curve $\pi \in \Pi(s_0)$ that visits segments $\tau(j)$, $j \in J'$ if and only if there is a feasible schedule σ' that can process all jobs in J'.* ∎

By this lemma, the problem is now to find a monotone curve that collects a maximum benefit set of segments $\tau(j)$.

3 Chart Graph

In this section, we define an edge-weighted acyclic digraph, called a *chart graph*, such that

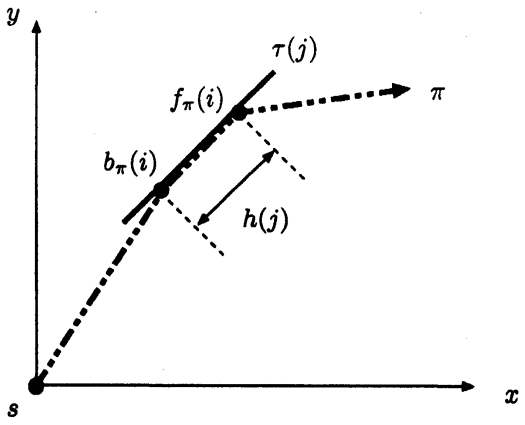

Figure 2: A monotone curve π visiting $\tau(j)$.

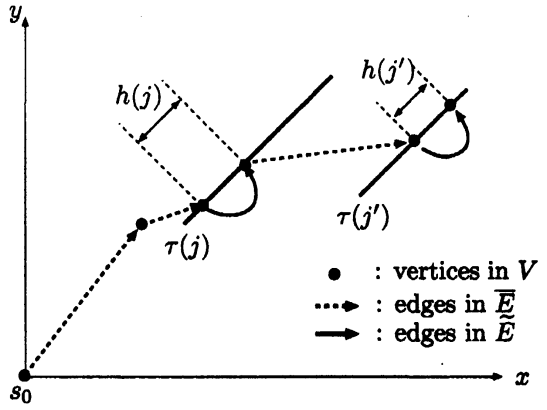

Figure 3: Illustration for a chart graph $G = (V, E = \overline{E} \cup \widetilde{E})$.

each directed path in a chart graph corresponds to a monotone curve $p \in \Pi(s_0)$. We assume that a given instance $I = (J, r, d, h, b, l)$ is represented as a set of segments $\tau(j)$ for all $j \in J$ in the x, y-plane.

A *chart graph* is a digraph $G = (V, E)$ such that its vertex set V is a finite subset of $R(s_0)$ with $s_0 \in V$ and its edge set E is a set of some ordered pairs (u, v) with $u, v \in V$ and $v \in R(u) - \{u\}$.

For each job j, let $\mathcal{E}_j = \{(q, q + h_j) \mid q \in \tau(j)\}$, which we regard as a set of edges from point q to $q + h_j$ for all $q \in \tau(j)$. Edge set E of a chart graph $G = (V, E)$ is partitioned into two subsets

$$\overline{E} = E \cap \bigcup_{j \in J} \mathcal{E}_j, \quad \widetilde{E} = E - \bigcup_{j \in J} \mathcal{E}_j$$

(see Fig. 3). Edge set E of a chart graph $G = (V, E)$ may be denoted by $E = \overline{E} \cup \widetilde{E}$. We easily see that any chart graph is acyclic.

Since each edge $(q, q') \in E$ satisfies $q' \in R(q)$, any directed path starting from the origin s_0 in G is a monotone curve starting from the origin s_0. Let $\Pi_G(s_0)$ denote the set of all directed path starting from the origin s_0 in G, where we may treat a path $p \in \Pi_G(s_0)$ as a monotone curve in the x, y-plane, i.e., $\Pi_G(s_0) \subseteq \Pi(s_0)$. A directed path in G visits segment $\tau(j)$ by using an edge $(q, q + h_j) \in \overline{E}$.

Let $\kappa(\pi)$ denote the total benefit of the segments collected by a monotone curve $\pi \in \Pi(s_0)$. Let $\nu(\pi)$ denote the number of segments collected by a monotone curve $\pi \in \Pi(s_0)$. Let π^* denote a curve that collects a maximum benefit set of segments of time windows over all monotone curves in $\Pi(s_0)$. Such a curve π^* is called an *optimal curve*. Let p^* denote a directed path in $\Pi_G(s_0)$ that collects a maximum benefit set of segments over all directed paths in G. Such a path p^* is called an *optimal path* in $\Pi_G(s_0)$. Note that $\kappa(p^*) > 0$ by assumption (1) and $J \neq \emptyset$. Define

$$\alpha(G) = \frac{\kappa(\pi^*)}{\kappa(p^*)}.$$

To find a path $\Pi_G(s_0)$ that visits many segments, we assign weight 0 for each edge $e \in \widetilde{E}$ and weight $b(j)$ for each edge $e \in \overline{E} \cap \mathcal{E}_j$, $j \in J$, and we define the weight $w(p)$ of a directed path p as the total weight of edges in p. Define

$$\beta(G) = \max\left\{ \frac{w(p)}{\kappa(p)} \mid p \in \Pi_G(s_0),\ \kappa(p) > 0 \right\}.$$

By definition of γ, we easily observe the following property.

Lemma 3.1 *Any directed path in G visits the same segment of a time window at most γ times (hence $\beta(G) \leq \gamma$).* ∎

The approximation ratio of a maximum weight path to an optimal curve is bounded as follows.

Theorem 3.2 *Let $\hat{p} \in \Pi_G(s_0)$ denote a maximum weight directed path in a chart graph G, and $\pi^* \in \Pi(s_0)$ be an optimal curve. Then $\kappa(\pi^*)/\kappa(\hat{p}) \leq \alpha(G)\beta(G)$.*

PROOF: Let $p^* \in \Pi_G(s_0)$ be an optimal path. Since $\hat{p}$ is a maximum weight path, we have $w(\hat{p}) \geq w(p^*)$, and since $w(p^*) \geq \kappa(p^*)$, we have $w(\hat{p}) \geq \kappa(p^*)$. On the other hand, from definition of $\alpha(G)$, we have $\kappa(p^*) = \kappa(\pi^*)/\alpha(G)$. Hence, we have $w(\hat{p}) \geq \kappa(\pi^*)/\alpha(G)$. Moreover, from definition of $\beta(G)$, we have $\kappa(\hat{p}) \geq w(\hat{p})/\beta(G)$. Therefore, it holds $\kappa(\hat{p}) \geq \kappa(\pi^*)/(\alpha(G)\beta(G))$. ∎

4 2-Approximate Path

This section gives how to construct from a given instance I a chart graph G with a moderate size that contains a 2-approximate path $p \in \Pi_G(s_0)$ for an optimal curve $\pi^* \in \Pi(s_0)$. Formally we will prove the next theorem.

Theorem 4.1 *Given an instance I, there exists a chart graph $G = (V, E)$ with $|V| = O(\gamma n), |E| = O(\gamma n^2)$ and $\alpha(G) \leq 2$.* ∎

Given an instance I, a chart graph in Theorem 4.1 can be constructed as follows. For each $j \in J$, let

$$\theta(j) = \lfloor (d(j) - r(j))/h_{\min} \rfloor,$$
$$c_i(j) = a(j) + ih^*,\ i = 0, 1, \ldots, \theta(j),$$
$$c_{\theta(j)+1}(j) = a'(j).$$

Note that $c_0(j) = a(j)$. For each $j \in J$, let $Q_j = \{c_i(j) \mid i = 0, 1, \ldots, \theta(j) + 1\}$, $Q'_j = \{q + h_j \mid q \in Q_j\}$. Let $Q = \bigcup_{j \in J} Q_j$ and $Q' = \bigcup_{j \in J} Q'_j$. For a point $z \in \mathbb{R}^2_+$, let $J(z) = \{j \in J \mid \tau(j) \cap R(z) \neq \emptyset\}$. Define the vertex and edge sets of chart graph $G_1 = (V_1, E_1)$ by

$$V_1 = \{s_0\} \cup Q \cup Q', \quad E_1 = \overline{E} \cup \widetilde{E},$$

where

$$\overline{E} = \bigcup_{j \in J} \{(q, q + h_j) \mid q \in Q_j\},$$

$$\widetilde{E} = \bigcup_{j \in J} \{(s_0, a(j))\} \cup \bigcup_{j' \in J(q)} \{(q, c_m(j')) \mid q \in Q', m = \min\{i \mid c_i(j') \succeq q\}\}.$$

Each edge $(q, q + h_j) \in \overline{E}$ is weighted by $b(j)$ and each edge in $\widetilde{E}$ is weighted by 0.

Figure 4 shows how an edge in $\widetilde{E}$ outgoing from a vertex $q \in Q$ on a segment $\tau(j)$ is constructed. In this example, $\widetilde{E}$ has no edges outgoing from q to any point on segment $\tau(j_5)$ since $\tau(j_5) \cap R(q) = \emptyset$. For $J(q) = \{j_2, j_3, j_4\}$ with $m = \min\{i \mid c_i(j_2) \succeq q\}$, $0 = \min\{i \mid c_i(j_3) \succeq q\}$, and $m' = \min\{i \mid c_i(j_4) \succeq q\}$, three edges $(q, c_m(j_2))$, $(q, c_0(j_3))$ and $(q, c_{m'}(j_4)) \in \widetilde{E}$ will be constructed.

We first consider the size of chart graph G_1.

Lemma 4.2 *For a chart graph $G_1 = (V_1, E_1)$, $|V_1| = \Theta(\gamma n)$ and $|E_1| = \Theta(\gamma n^2)$.*

PROOF: Since $|Q_j| = |Q'_j| \leq \theta(j) + 2$, $j \in J$, and $\max_{j \in J} \theta(j) = \lfloor \max_{j \in J}(d(j) - r(j))/\min_{j \in J} h(j) \rfloor = \gamma - 1$, we have $|Q| = ||Q'| = \Sigma_{j \in J}(\theta(j) + 2) = O(n\gamma)$. From this, we have $|V_1| = O(\gamma n)$ and $|\overline{E}| = |Q'| = O(\gamma n)$. Since G_1 has at most one edge in $\widetilde{E}$ outgoing from each point $q \in Q'$ to a point on a segment $\tau(j)$, G_1 has at most n edges in $\widetilde{E}$ outgoing from each $q \in Q'$. Hence, $|\widetilde{E}| = O(n|Q'|) = O(\gamma n^2)$. Therefore, $|E_1| = |\overline{E}| + |\widetilde{E}| = O(\gamma n^2)$.

We can construct an instance I such that chart graph G_1 has $|V_1| = \Omega(\gamma n)$ vertices and $|E_1| = \Omega(\gamma n^2)$ edges (the detail is omitted due to space limitation). ∎

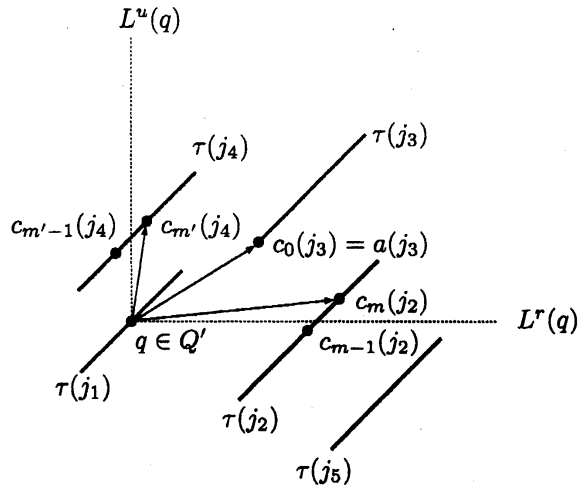

Figure 4: Illustration of edges $(q, c_m(j)) \in \widetilde{E}$ outgoing from a point $q \in Q'$ to a point $c_m(j)$ on a segment $\tau(j)$, $j \in J(q)$.

We next prove that $\alpha(G_1) \leq 2$.

Lemma 4.3 *Let G_1 be a chart graph constructed from an instance I in the above manner. For an arbitrary curve $\pi \in \Pi(s_0)$, G_1 has a path $p \in \Pi_G(s_0)$ such that $\kappa(p) \geq \kappa(\pi)/2$.*

PROOF: Given an arbitrary curve $\pi \in \Pi(s_0)$, we number the segments collected by π so that segment $\tau(i)$ denotes the ith segment collected by π.

We proceed by an induction on i. We assume that, for the finishing point $f_\pi(i)$, G_1 has a path $p \in \Pi_G(s_0)$ such that

(A1) p reaches a point $s' \in Q'$ with $f_\pi(i) \succeq s'$,

(A2) $\kappa(p) \geq \frac{1}{2} \sum_{k=1}^{i} b(k)$.

Assumptions (A1) and (A2) hold for $i = 1$ as follows. Since G_1 has two edges $(s_0, c_0(1)) \in \widetilde{E}$ and $(c_0(1), c_0(1) + h_1) \in \overline{E}$, a path consisting of these two edges collects $\tau(1)$ and reaches point $c_0(1) + h_1$ after collecting $\tau(1)$. Since this path

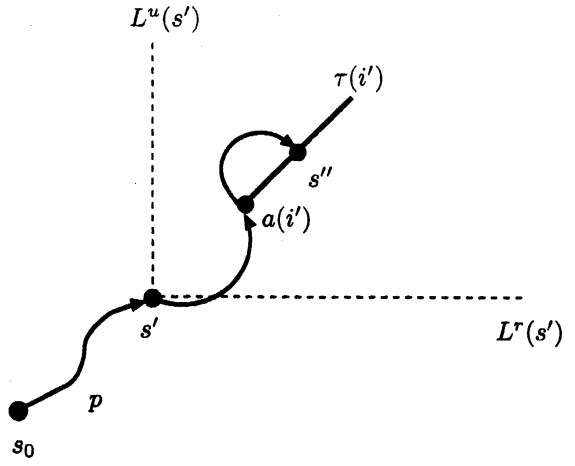

Figure 5: Case 2-(a) in the proof of Lemma 4.3.

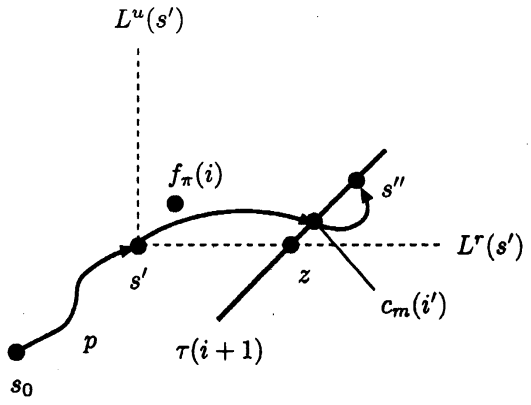

Figure 6: Case 2-(b) in the proof of Lemma 4.3.

starts to collect $\tau(1)$ from its endpoint $c_0(1)$, we have $f_\pi(1) \succeq c_0(1)+h_1$. Moreover, $c_0(1)+h_1 \in Q'$.

We now assume that (A1) and (A2) hold for some $i < \nu(\pi)$, and extend the endpoint s' of path p in G_1 to another point $s'' \in Q'$. Since π collects all segments in $\{\tau(k) \mid i+1 \le k \le \nu(\pi)\}$, we have $R(f_\pi(i)) \cap \tau(k) \neq \emptyset$ for each $i+1 \le k \le \nu(\pi)$, and since $f_\pi(i) \succeq s'$, we have $R(s') \cap \tau(k) \neq \emptyset$.

Case 1 $\nu(\pi) = i+1$. Since $s' \in Q'$ and $R(s') \cap \tau(i+1) \neq \emptyset$, G_1 has an edge $(s', c_m(i+1)) \in \widetilde{E}$ where $m = \min\{i \mid c_i(i+1) \succeq s'\}$. Moreover, G_1 has an edge $(c_m(i+1), c_m(i+1)+h_{i+1}) \in \overline{E}$. The path $p' \in \Pi_G(s_0)$ obtained from p by adding these two edges collects $\tau(i+1)$. By induction hypothesis, we have $\kappa(p') \ge \frac{1}{2}\sum_{k=1}^{i+1} b(k)$, proving the lemma.

Case 2 $\nu(\pi) \ge i+2$. Let $i' \in \{i+1, i+2\}$ be such that $b(i') = \max\{b(i+1), b(i+2)\}$.

(a) $\tau(i') \subset R(s')$ (see Fig. 5). Since π collects $\tau(i')$, we have $f_\pi(i+2) \succeq c_0(i')+h_{i'}$. Since $s' \in Q$, G_1 has two edges $(s', c_0(i')) \in \widetilde{E}$ and $(c_0(i'), c_0(i')+h_{i'}) \in \overline{E}$. Then the path $p' \in \Pi_G(s_0)$ obtained from p by adding these two edges reaches point $s'' = c_0(i') + h_{i'}$ after collecting $\tau(i')$. Hence the resulting path p' satisfies

$$\kappa(p') \ge \frac{1}{2}\sum_{k=1}^{i} b(k) + b(i') \ge \frac{1}{2}\sum_{k=1}^{i+2} b(k),$$

implying that (A2) holds for $i+2$. Moreover, points $s'' = c_0(i')+h_{i'} \in Q'$ and $f_\pi(i+2)$ satisfy condition (A1) for $i+2$.

(b) $\tau(i') \not\subset R(s')$ (see Fig. 6). In this case, segment $\tau(i')$ intersects $L^r(s')$ or $L^u(s')$ since $R(s') \cap \tau(i') \neq \emptyset$; We assume without loss of generality that $\tau(i')$ intersects $L^r(s')$ at a point z. Let $m = \min\{i \mid c_i(i') \succeq s'\}$. Since G_1 has two edges $(s', c_m(i')) \in \widetilde{E}$ and $(c_m(i'), c_m(i')+h_{i'}) \in \overline{E}$, the path $p' \in \Pi_G(s_0)$ obtained from p by adding these two edges reaches point $s'' = c_m(i')+h_{i'}$ after collecting $\tau(i')$. Hence the resulting path p' satisfies $\kappa(p') \ge \frac{1}{2}\sum_{k=1}^{i} \kappa(k) + b(i') \ge \sum_{k=1}^{i+2} \kappa(k)$, implying that (A2) holds for $i+2$. Since z is an intersection point of segment $\tau(i')$ with $L^r(s')$ and $f_\pi(i) \succeq s'$, we have $f_\pi(i+2) \succeq z+h_{i+1}+h_{i+2}$. On the other hand, since $z + h_{\min} \succeq c_m(i')$, we have $z+h_{\min}+h_{i'} \succeq c_m(i')+h_{i'}$. Hence, we finally have $f_\pi(i+2) \succeq c_m(i') + h_{i'} = s''$, implying condition (A2) for $i+2$. ∎

Lemmas 4.2 and 4.3 prove Theorem 4.1.

5 Computing a 2-approximate Solution

In this section, we describe an algorithm for finding a directed path in $G_1 = (V_1, E_1)$ that visits a maximum benefit set of segments $\tau(j)$. We call a directed path which does not visit the same segment more than once a *non-overlapping path*.

We apply dynamic programming to compute a maximum benefit non-overlapping path in G_1. By definition of ρ, we have the following lemma.

Lemma 5.1 *Let $\pi \in \Pi(s_0)$ denote a monotone curve, and let $\tau(i)$ denote the ith segment collected by π. If $\nu(\pi) \le \rho$ or $\tau(k) \notin \{\tau(k-1), \tau(k-2), \ldots, \tau(k-\rho-1)\}$ for each $\rho+1 \le k \le \nu(\pi)$, then π is a non-overlapping path.* ∎

By Lemma 5.1, a dynamic programming algorithm computes a maximum weight non-overlapping path by maintaining lists of last $\rho+1$ segments collected by curves.

Theorem 5.2 *For an instance I to the MAX-VSP-PATH, a 2-approximation solution can be obtained in $O(\gamma n^{(3+\rho)})$ time.* ∎

For sparse instances, by modifying G_1, we can obtain a 2-approximation solution by computing a maximum weight path p in the modified acyclic digraph.

Theorem 5.3 *For a sparse instance I to the MAX-VSP-PATH, a 2-approximate solution can be obtained in $O(\gamma n^2)$ time.* ∎

6 Concluding Remarks

In this paper, we designed an approximation algorithm for the MAX-VSP-PATH with general time windows, nonzero handling times and general benefits. For this, we regarded the MAX-VSP-PATH as the problem of finding a monotone curve that collects a maximum benefit set of segments in the x, y-plane. We then introduced a chart graph to approximate an optimal monotone curve by a directed path. Based on this chart graph, we introduced approximation algorithms. We gave an $O(\gamma n^{(3+\rho)})$ time 2-approximation algorithm, and gave an $O(\gamma n^2)$ time 2-approximation algorithm for sparse instances. We proved that the MAX-VSP-PATH is weakly NP-complete even for sparse instances with nonzero handling times. It is left as a remaining task to analyze the problem complexity of the MAX-VSP-PATH when ρ is constant. We remark that the idea of reducing the problem to an monotone curve problem has been used by Bar-Yehuda et al. [2] to design approximation algorithms for the MAX-VSP-PATH with no handling time. However, our method of constructing a digraph that approximates an optimal monotone curve is completely different from their digraphs. In fact, our chart graph cannot handle instances with zero handling time, and it is important to investigate a common generalization of these two ways of approximating monotone curves by digraphs.

References

[1] Y. Asahiro, T. Horiyama, K. Makino, H. Ono, T. Sakuma and M. Yamashita, "How to collect balls moving in the euclidean plane," Proc. Computing: The Australasian Theory Symposium (CATS), pp.1-16, Jan. 2004. (Electronic Notes in Theoretical Computer Science, vol. 91, pp. 229-245.)

[2] R. Bar-Yehuda, G. Even and S. Shahar, "On approximating a geometric prize-collecting traveling salesman problem with time windows," In *Lecture Notes in Computer Science*, vol. 2832, pp. 55-66, Springer-Verlag, Sept. 2003.

[3] L. Bodin, B. Golden, A. Assad and M. Ball, "Routing and scheduling of vehicles and crews: the state of the art," *Computers & Operations Research*, vol. 10, pp. 62-212, 1983.

[4] L. D. Bodin, "Twenty years of routing and scheduling," *Operations Research*, vol. 39, pp.571-579, 1990.

[5] M. R. Garey and D. S. Johnson, "Two-processor scheduling with start times and deadlines," *SIAM Journal on Computing*, vol. 6, pp. 416-426, 1977.

[6] Y. Karuno, H. Nagamochi and T. Ibaraki, "Better approximation ratios for the single-vehicle scheduling problems on line-shaped networks," *Networks*, vol. 39, no. 4, pp. 203-209, 2002.

[7] Y. Karuno and H. Nagamochi, "2-Approximation algorithms for the multi-vehicle scheduling problem on a path with release and handling times," *Discrete Applied Mathematics*, vol. 129, pp. 433-447, 2003.

[8] Y. Karuno and H. Nagamochi, "Scheduling vehicles on trees," *Pacific Journal of Optimization*, vol. 1, pp. 527-543, 2005.

[9] H. Psaraftis, M. M. Solomon, L. Magnanti and T .U. Kim, "Routing and scheduling on a shoreline with release times," *Management Science*, vol. 36, pp.212-223, 1990.

[10] J. N. Tsitsiklis, "Special cases of traveling salesman and repairman problems with time windows," *Networks*, vol. 22, pp. 263-282, 1992.

[11] G. Young and C. Chan, "Single-vehicle scheduling with time window constraints," *J. Scheduling*, vol. 2, pp.175-187, 1999.