



ELSEVIER

Available at

[www.ElsevierComputerScience.com](http://www.ElsevierComputerScience.com)

POWERED BY SCIENCE @ DIRECT®

Electronic Notes in  
Theoretical Computer  
Science

Electronic Notes in Theoretical Computer Science 91 (2004) 229–245

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

## How to Collect Balls Moving in the Euclidean Plane

Yuichi Asahiro<sup>a,1</sup>, Takashi Horiyama<sup>c,2</sup>, Kazuhisa Makino<sup>b,3</sup>,  
Hirotaka Ono<sup>d,4</sup>, Toshinori Sakuma<sup>d,4</sup>, Masafumi Yamashita<sup>d,4</sup>

<sup>a</sup> *Department of Social Information Systems, Faculty of Information Science, Kyushu Sangyo University, Fukuoka 813-8503, Japan.*

<sup>b</sup> *Division of Systems Science, Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka 560-8531, Japan.*

<sup>c</sup> *Department of Communications and Computer Engineering, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan*

<sup>d</sup> *Department of Computer Science and Communication Engineering, Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka 812-8581, Japan.*

---

### Abstract

In this paper, we study how to collect  $n$  balls moving with constant velocities in the Euclidean plane by  $k$  robots moving on straight track-lines through the origin. Since all the balls might not be caught by robots, differently from Moving-Target TSP, we consider the following 3 problems in various situations: (i) deciding if  $k$  robots can collect all  $n$  balls, (ii) maximizing the number of the balls collected by  $k$  robots, and (iii) minimizing the number of the robots to collect all  $n$  balls. The situations considered here contain the cases in which track-lines are given (or not), and track-lines are identical (or not). For all problems and situations, we provide polynomial time algorithms or proofs of intractability, which clarify the tractability-intractability frontier in the ball collecting problems in the Euclidean plane.

**Keywords:** Moving-Target TSP, Vehicle routing problem, Partially ordered set, Combinatorial optimization.

---

<sup>1</sup> Email: [asahiro@is.kyusan-u.ac.jp](mailto:asahiro@is.kyusan-u.ac.jp)

<sup>2</sup> Email: [horiyama@i.kyoto-u.ac.jp](mailto:horiyama@i.kyoto-u.ac.jp)

<sup>3</sup> Email: [makino@sys.es.osaka-u.ac.jp](mailto:makino@sys.es.osaka-u.ac.jp)

<sup>4</sup> Email: [ono@csce.kyushu-u.ac.jp](mailto:ono@csce.kyushu-u.ac.jp), [sakuma@tcslab.csce.kyushu-u.ac.jp](mailto:sakuma@tcslab.csce.kyushu-u.ac.jp), [mak@csce.kyushu-u.ac.jp](mailto:mak@csce.kyushu-u.ac.jp)

# 1 Introduction

We consider the problem of collecting  $n$  balls (or objects) moving around by using  $k$  robots. We often encounter this kind of situation as real-world applications (e.g., [8,9]). For example, supply ships resupply patrolling boats in the sea, and aircraft must intercept a number of mobile ground units. In the first example, boats and ships correspond to balls and robots, respectively. The problem can be seen as a natural extension of the classical Traveling Salesman Problem (TSP) and was studied under the names of Moving-Target TSP [9] and Kinetic TSP [8].

In the classical TSP, we are given a set of balls (or cities) associated with the distances between each pair of balls. The objective is to compute a shortest tour that collects all the balls by a robot (or visits all the cities by a salesman). In the Moving-Target TSP, the balls collected by a robot are not stationary but moving around. Clearly, Moving-Target TSP is NP-hard, since classical TSP is a special case of Moving-Target TSP in which the velocities of all objects are permanently 0.

Helvig, et al. [9] investigated this problem when each ball  $b_i$  moves at a given constant velocity  $\mathbf{v}_i$  from an initial position  $p_i \in \mathbf{R}^d$ , and a robot starts at the origin with the maximum speed  $v > |\mathbf{v}_i|$ . They showed that the problem can be solved in  $O(n^2)$  time if  $d = 1$  (i.e., the robot and all balls are confined to a single line), and can be approximated in polynomial time with ratio  $(1 + \alpha)$ , where  $\alpha$  is an approximation ratio of an arbitrary classical TSP heuristic, if  $O(\log n / \log \log n)$  balls are moving. They also considered the case in which multi-robots can be used. Hammar and Nilsson [8] studied the 2-dimensional Euclidean case. For example, it can be shown that, if all the balls move with the same velocity, the problem has a PTAS, but in general, it cannot be approximated better than by a factor of  $2^{\Omega(\sqrt{n})}$  in polynomial time, unless  $P=NP$ .

In the Moving Target TSP (or kinetic TSP), it is assumed that any robot can move faster than any ball, and hence robots can always collect all balls. However, in some scenarios (e.g., [4,15]), it is natural to consider the case in which robots cannot collect all balls. This situation results from the fact that robots cannot move faster than balls and/or robots can move only in the restricted spaces such as lines. In this situation, the following fundamental problems have to be studied.

- (i) deciding if given  $k$  robots can collect given  $n$  balls, and if yes, computing such a robot schedule.
- (ii) computing a robot schedule that maximizes the number of the balls collected by given  $k$  robots.

- (iii) computing a robot schedule that minimizes the number of the robots collecting given  $n$  balls.

It is clear that (ii) and (iii) are natural extensions of (i). We generically call these kinds of problems the *Ball Collecting Problems (BCPs)*.

This paper investigates the BCPs in the Euclidean plane, under the assumption that each robot can move only on a straight line. More formally, our problems can be defined as follows.

Given a set  $B = \{b_1, b_2, \dots, b_n\}$  of balls in the Euclidean plane (with  $x$ - and  $y$ -axes), each  $b_i$  moving at a constant velocity  $\mathbf{v}_i = (v_i \cos \varphi_i, v_i \sin \varphi_i)$  from an initial point  $p_i^{(0)} = (x_i^{(0)}, y_i^{(0)}) \in \mathbb{R}^2$ , and a set  $R = \{r_1, r_2, \dots, r_k\}$  of (homogeneous) robots, each  $r_j$  starting from the origin (or depot)  $o = (0, 0)$  and moving with maximum speed  $v > 0$  on a line  $\ell_j$  through the origin  $o$ , compute robot schedules described in (i), (ii) and (iii).

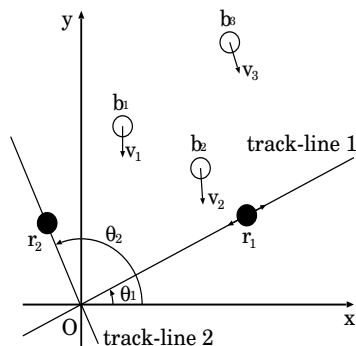


Fig. 1. BCP with 2 robots on 2 track-lines

Fig. 1 illustrates our problems, when 2 robots move on 2 different lines. We also consider the case in which lines  $\ell_j$  are not predetermined (i.e., we can decide an angle  $\theta_j$  of line  $\ell_j$  as a part of the scheduling). Note that the assumption that each ball moves at a constant velocity has been studied in [8,9] (see [8,9] for the motivations). The assumption that each robot moves on a line is motivated as follows. First, it is reasonable to consider a kind of symmetric condition between balls and robots. Both move on straight lines. Second, a line is one of the most fundamental tracks of a robot. Actually, many kinds of tracks of a robot can be decomposed into several lines. This means that the results under the assumption above are useful for general shapes of tracks. Finally, the assumption is often encountered in the real world, e.g., railway, bridge and ropeway.

We remark that our problems when  $n$  robots all move on a given single line  $\ell$  can be regarded as new models of the *Vehicle Scheduling Problem (VSP)* on

line-shaped network (e.g., [10,14,2,16]). VSPs are classified by the conditions on release times, handling times, due times (or deadlines), and the objectives such as minimizing the completion time and minimizing the maximum lateness. Robots and catching balls in our case are regarded as vehicles and jobs, respectively. We consider only the balls which crosses line  $\ell$  exactly once (see the discussion in Section 2.1). The handling time of each job is defined as 0, and the release time and the due time of job  $b_i$  are defined as the time when it crosses line  $\ell$ . Each job  $b_i$  is located at the position where it crosses line  $\ell$ . Then we can see that problem (i) is to decide whether the maximum lateness of this VSP is 0. This kind of setting is well studied. However, the objectives corresponding to problems (ii) and (iii) (for example, to compute a schedule of vehicles that maximizes the number of the jobs done by their due times) are not investigated as the VSPs.

We extensively study the BCPs defined above mainly from computational points of view. For all problems, polynomial time algorithms or proofs of intractability are provided (although it remains important to improve efficiency of the algorithms for polynomially solvable cases and to develop polynomial approximation algorithms for NP-hard cases). Thus, our work gives a complete picture of the tractability-intractability frontier in the 2-dimensional Euclidean BCP when balls  $b_i$  move with fixed constant velocities  $\mathbf{v}_i$  and homogeneous robots  $r_j$  move lines  $\ell_j$  through the depot  $o$ . A summary of the results in this paper is given in Table 1.

We first consider the case in which  $n$  robots all move on a given single line  $\ell$ , i.e.,  $\ell_j = \ell$  for all  $j$ . In this case, we present an  $O(n \log n)$  time algorithm for the problem (iii) (and hence (i) is solvable in  $O(n \log n)$  time). This is optimal in the sense that the problem contains the sorting problem as a special case, where the sorting problem is known to need  $\Omega(n \log n)$  time for the comparison-based algorithms. Moreover, we show that the problem (ii) is solvable in  $O(n \log n)$  time, if  $k = 1$ , and  $O(n^3)$  time if  $k \geq 2$ . The result when  $k = 1$  is also optimal in the sense above.

On the other hand, if given lines  $\ell_j$  are not identical, most problems seems to be intractable. Note that the case of  $k = 1$  is studied in the single line case. We show that the problem (i) can be solved in  $O(n^2)$  time when  $k = 2$ , however, it is NP-complete when  $k \geq 3$ . The problems (ii) and (iii) are both NP-hard.

We finally study the case in which lines  $\ell_j$  are not given (i.e., we can decide an angle  $\theta_j$  of line  $\ell_j$  as a part of the scheduling). We show that  $\Theta(n^2)$  lines are essentially different for our problems, although there exist infinitely many lines through the origin. By combining this with the previous positive results, we provide polynomial time algorithms. For example, if we choose a single

line  $\ell$  for all robots, then the problem (iii) is solvable in  $O(n \log n) \times O(n^2) = O(n^3 \log n)$  time, and if we can choose distinct lines for each robot, the problem (i) when  $k = 2$  can be solved in  $O(n^2) \times \binom{O(n^2)}{2} = O(n^6)$  time. We show that the problem (i) when  $k = 1$  can be solved in  $O(n^2 \log n)$  time, which improves upon the simple combined algorithm. Furthermore, we show the intractability of the problems does not depend on the fact that lines are given or not.

Table 1  
Summary of results.

Lines $\ell_j$		(i)		
		$k = 1$	$k = 2$	$k \geq 3$
Input	single	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
	multiple	$O(n \log n)$	$O(n^2)$	NP-complete
Output	single	$O(n^2 \log n)$	$O(n^3 \log n)$	$O(n^3 \log n)$
	multiple	$O(n^2 \log n)$	$O(n^6)$	NP-complete
Lines $\ell_j$		(ii)		(iii)
		$k = 1$	$k \geq 2$	
Input	single	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
	multiple	$O(n \log n)$	$O(n^2)$	NP-complete
Output	single	$O(n^2 \log n)$	$O(n^3 \log n)$	$O(n^3 \log n)$
	multiple	$O(n^2 \log n)$	$O(n^6)$	NP-complete

Input and Output mean that lines  $\ell_j$  are chosen as a part of input and output, respectively.

The rest of the paper is organized as follows. In Sections 2 and 3, we investigate the BCPs when the track-lines  $\ell_j$  are given in advance. Section 2 deals with the case in which all  $\ell_j$  are identical, and Section 3 deals with the general case. Section 4 considers the BCPs when  $\ell_j$  are not given. Section 5 concludes this paper and discusses some possible extensions of the BCPs.

## 2 BCPs with a given single track-line

In this section, we consider the BCPs when all  $k$  robots move on the identical line  $\ell$ .

### 2.1 Single robot BCP

We start with the most fundamental situation that a single robot tries to collect all  $n$  balls. We first exclude two kinds of balls: (1) the balls which do not cross the track-line  $\ell$  and (2) the balls which move on  $\ell$ . Note that the robot never catches the balls of type (1). Let  $b_i$  be a ball of type (2). If  $b_i$

goes away from the origin with the speed  $v_i > v$ , then the robot cannot catch up with it; Otherwise, it is always caught by the robot. Hence we assume without loss of generality that all balls cross  $\ell$  exactly once.

Let  $t_i$  denote the time when ball  $b_i$  crosses the line  $\ell$ , and let  $p_i = (x_i, y_i)$  denote the point where  $b_i$  crosses  $\ell$ . Define  $d_i = \text{dist}(o, p_i)$  if  $x_i \geq 0$ , and  $d_i = -\text{dist}(o, p_i)$ , otherwise, where  $\text{dist}(o, p_i)$  denotes the Euclidean distance between the origin  $o$  and  $p_i$ .<sup>5</sup> It is clear that if the robot can catch all balls, then the balls have to be caught in the order of  $t_i$ . Therefore, the problem (i) can be solved by first sorting  $t_i$  ( $i = 1, \dots, n$ ) and then checking if

$$v(t_{p_{k+1}} - t_{p_k}) \geq |d_{p_{k+1}} - d_{p_k}|, \quad k = 1, 2, \dots, n-1,$$

where  $t_{p_1} \leq t_{p_2} \leq \dots \leq t_{p_n}$ . This immediately implies the following result.

**Theorem 2.1** It can be checked in  $O(n \log n)$  time if all balls can be collected by a robot moving on a given track-line. Furthermore, if so, such a robot schedule can be computed in  $O(n \log n)$  time.  $\square$

In the following sections, we show that the problem (ii), which is a generalization of the problem (i), is also solvable in  $O(n \log n)$  time. This is optimal in the sense discussed in the introduction.

Before describing an optimal algorithm, we first present a simple  $O(n^2)$  time algorithm for the problem (ii) in order to understand the optimal one easily. The algorithm makes use of a directed acyclic graph (DAG) constructed from  $d_i$  and  $t_i$  above.

### 2.1.1 DAG based algorithm

Let  $\theta$  denote the angle of a track-line  $\ell$ . A directed graph  $G(\theta) = (V = \{1, 2, \dots, n\}, A = \bigcup_{i=0}^n A_i)$  is defined by

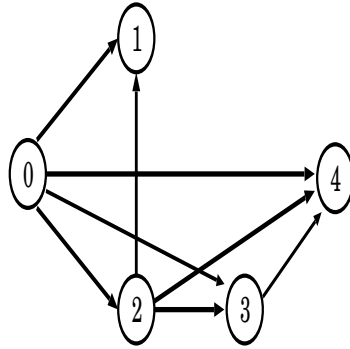
$$A_0 = \{(0, i) \mid \text{The robot starting at the origin } o \text{ can catch ball } b_i\}$$

$$A_i = \{(i, j) \mid \text{The robot can catch ball } b_j \text{ after catching } b_i\}, i = 1, 2, \dots, n.$$

It is easy to see that  $(0, i) \in A_0$  if and only if  $v \cdot t_i \geq d_i$ , and  $(i, j) \in A_i$  for  $i \geq 1$  if and only if  $v \cdot (t_j - t_i) \geq |d_j - d_i|$ . For example, Fig. 2 shows the situation that all balls can be caught directly by the robot starting at the origin  $o$  (since  $G(\theta)$  contains arcs  $(0, 1)$ ,  $(0, 2)$  and  $(0, 3)$ ), ball  $b_1$  can be caught after catching  $b_2$  (since  $G(\theta)$  contains arc  $(2, 1)$ ), but  $b_3$  cannot be caught after catching  $b_1$  (since  $G(\theta)$  does not contain arc  $(1, 3)$ ), and so on.

Note that  $G(\theta)$  is transitive, i.e., if  $(i_1, i_2), (i_2, i_3) \in A$  then  $(i_1, i_3) \in A$ . If  $G(\theta)$  contains a cycle  $C$ , then  $C$  does not contain node 0, since no arc enters node 0, and all nodes in  $C$  have the same  $d_i$  and  $t_i$  (i.e.,  $d = d_i$  and  $t = t_i$

<sup>5</sup> If  $\ell$  coincides with  $y$ -axis,  $d_i = \text{dist}(o, p_i)$  if  $y_i \geq 0$ , and  $d_i = -\text{dist}(o, p_i)$  otherwise.

Fig. 2. An example of a directed graph  $G(\theta)$ 

for all  $i \in C$ ), since  $i, j \in C$  implies  $(i, j), (j, i) \in A$  by the transitivity of  $A$ , which further implies  $t_j - t_i, t_i - t_j \geq |d_j - d_i|/v (\geq 0)$ . Moreover, we can see the following proposition.

**Proposition 2.2** Any directed path from node 0 in  $G(\theta)$  represents feasible robot schedules. Conversely, any feasible robot schedule can be represented by a directed path from node 0. Furthermore, the length of the path in  $G(\theta)$  represents the number of the balls collected under the corresponding schedules.  $\square$

We now construct the directed acyclic graph (DAG)  $G^*(\theta) = (V^*, A^*)$  from  $G(\theta)$  by contracting every strongly connected component in it. Here node  $0^*$  in  $V^*$  corresponding to 0 in  $V$  has a weight  $w(0^*) = 0$ , and each node  $i^* (\neq 0^*)$  in  $V^*$  has a weight  $w(i^*)$  representing the number of the nodes in  $V$  contracted into  $i^*$ . Then the problem (ii) (i.e., maximizing the number of the collected balls) can be solved by finding a longest (or critical) path starting from  $0^*$  (corresponding to 0 in  $G(\theta)$ ) in this DAG  $G^*(\theta)$ , where the length of a path  $P$  is the weighted sum of the nodes in  $P$ . This problem is also studied in PERT as one of the scheduling problems, and is solvable in  $O(|V^*| + |A^*|)$  time (e.g., [11]). Since  $|V| \leq n$ ,  $|A^*| \leq n^2$ , and  $G^*(\theta)$  can be constructed in  $O(n^2)$  time, the problem (ii) is solvable in  $O(n^2)$  time.

**Theorem 2.3** We can compute in  $O(n^2)$  time a robot schedule that maximizes the number of the balls collected by a robot moving on a given track-line.  $\square$

One might see that the size of  $G^*(\theta)$  is small if it is constructed from a problem instance of the BCP. Unfortunately, this is not the case. We have a problem instance such that  $|A^*| = \Omega(n^2)$ . More precisely, even if we remove from  $A^*$  all transitive arcs  $a$  (i.e.,  $a = (i_1, i_3)$  if  $(i_1, i_2), (i_2, i_3) \in A^*$ ), the size of arcs is still  $\Omega(n^2)$ . This means that all the algorithms constructing  $G^*$

explicitly require  $\Omega(n^2)$  time. Therefore, we need a method to represent  $G^*$  (or  $G$ ) implicitly, in order to construct a faster algorithm.

### 2.1.2 Chart based algorithm

This section introduces a chart  $H(\theta)$  to represent  $G(\theta)$  implicitly, and based on this chart, we present an  $O(n \log n)$  time algorithm for the problem (ii) when a single robot moves on a given track-line. We first consider a 2-dimensional chart  $F(\theta)$  with  $t$ - and  $d$ -axes.  $F(\theta)$  has  $n + 1$  points  $(t_i, d_i)$ ,  $i = 0, 1, \dots, n$ , associated with the region  $R_i$  defined by the following 2 inequalities:

$$\begin{aligned} d &\leq v(t - t_i) + d_i \\ d &\geq -v(t - t_i) + d_i. \end{aligned}$$

Here  $(t_0, d_0) = (0, 0)$ , representing the origin  $o$  in the (original) Euclidean plane  $\mathbf{R}^2$ . Fig. 3 shows an example of  $F(\theta)$ . In this chart, a point  $(t, d)$  represents a state that a robot is located at point  $d$  at time  $t$ . Therefore, a point  $(t, d)$  is contained in  $R_i$  if and only if a robot starting at state  $(t_i, d_i)$  can reach the state  $(t, d)$ . For example,  $(t_1, d_1) \in R_2 \subseteq R_0$  in Fig. 3 means that robot can catch ball  $b_1$  after moving from the origin at time 0 and even after catching ball  $b_2$ .

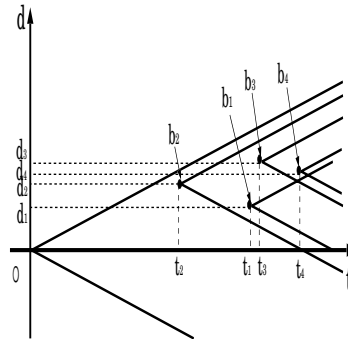


Fig. 3. A chart  $F(\theta)$  corresponding to the directed graph  $G(\theta)$  in Fig. 2

We now define a chart  $H(\theta)$  with  $\alpha$ - and  $\beta$ -axes to understand this relations easily. For each  $i$ , we define  $\alpha_i$  and  $\beta_i$  by  $\alpha_i = (vt_i - d_i)/(2v)$  and  $\beta_i = (vt_i + d_i)/2v$ ;  $t$  values of the points at which lines  $d = -vt$  (resp.,  $d = vt$ ) and  $d = v(t - t_i) + d_i$  (resp.,  $d = -v(t - t_i) + d_i$ ). Chart  $H(\theta)$  has  $n + 1$  points  $(\alpha_i, \beta_i)$ ,  $i = 0, 1, \dots, n$ . Here we identify the points  $(\alpha_i, \beta_i)$  ( $i = 1, 2, \dots, n$ ) with balls  $b_i$ . Fig. 4 shows an example of  $H(\theta)$  (corresponding to  $F(\theta)$  in Fig. 3).

For two points  $(\alpha_i, \beta_i)$  and  $(\alpha_j, \beta_j)$ , we write  $(\alpha_i, \beta_i) \leq (\alpha_j, \beta_j)$  if  $\alpha_i \leq \alpha_j$  and  $\beta_i \leq \beta_j$ , and  $(\alpha_i, \beta_i) < (\alpha_j, \beta_j)$  if  $(\alpha_i, \beta_i) \leq (\alpha_j, \beta_j)$  and  $(\alpha_i, \beta_i) \neq (\alpha_j, \beta_j)$ .



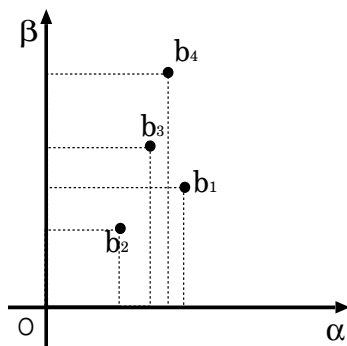


Fig. 4. A chart  $H(\theta)$  corresponding to  $F(\theta)$  in Fig. 3

**Lemma 2.4** Let  $\alpha_i$  and  $\beta_i$  be defined as above. Then we have the following necessary and sufficient conditions.

- (i) Ball  $b_i$  can be caught by a robot starting at origin  $o$  if and only if  $(0, 0) \leq (\alpha_i, \beta_i)$  holds.
- (ii) Ball  $b_j$  can be caught by a robot after catching ball  $b_i$  if and only if  $(\alpha_i, \beta_i) \leq (\alpha_j, \beta_j)$  holds.  $\square$

**Corollary 2.5** Given a chart  $H(\theta)$ , we can construct  $G(\theta)$  (and hence  $G^*(\theta)$ ) by connecting arcs from  $i$  to  $j$  if  $(\alpha_i, \beta_i) \leq (\alpha_j, \beta_j)$ .  $\square$

For example, we can see that Fig. 2 can be constructed from Fig. 4. Our algorithm uses  $H(\theta)$  instead of  $G^*(\theta)$ . Before showing how to use this  $H$ , let us consider the dual of the problem (ii).

Note that a transitive DAG  $G^*(\theta) = (V^*, A^*)$  can be regarded as a partially ordered set (*poset*, in short)  $P(\theta) = (V^*, \preceq)$ , where  $i^* \preceq j^*$  if either  $i^* = j^*$  or  $(i^*, j^*) \in A^*$ . Let us recall that a *chain* (resp., *antichain*) in a poset is a set of pairwise comparable (resp., incomparable) elements. Then the following theorem is well-known, called a *weighted version of the polar (or dual) Dilworth theorem* [12].

**Theorem 2.6** Let  $P$  be a finite poset with a nonnegative integer weight  $w(e)$  of each element  $e$ . Then the maximum weight of a chain is equal to the minimum number of antichains covering all elements  $e$  in  $P$  by  $w(e)$  times.  $\square$

It follows from Proposition 2.2 and the subsequent discussion that the problem (ii) is to compute a maximum weighted chain of  $P$ . Therefore, by Theorem 2.6, the dual of the problem (ii) is to compute a minimum  $w$ -multiple antichain covering of  $P$ . We construct an optimal algorithm for the problem (ii) by solving this dual problem with the help of chart  $H$ .

Given a set of balls (or points)  $S$  in chart  $H(\theta)$ ,  $b_i$  in  $S$  is called *pareto-optimal* if no  $b_j \in S$  satisfies  $(\alpha_j, \beta_j) > (\alpha_i, \beta_i)$ , and  $S' \subseteq S$  is called a *pareto-*

optimal set of  $S$  if all balls in  $S'$  are pareto-optimal and no two balls  $b_i$  and  $b_j$  in  $S'$  satisfy  $(\alpha_i, \beta_i) = (\alpha_j, \beta_j)$ . Note that a maximal pareto-optimal set is always maximum, and pareto-optimal sets correspond to antichains of  $P(\theta)$ . Moreover, a minimum  $w$ -multiple antichain covering of  $P(\theta)$  can be computed by the following algorithm which finds (and removes) maximum pareto-optimal sets of  $B$  repeatedly.

**Algorithm** ANTICHAIN-PARTITION

**Input:** A set of balls  $B = \{b_1, b_2, \dots, b_n\}$ .

**Output:** A minimum antichain partition  $(S_1, S_2, \dots, S_r)$  of  $B$  (corresponding to a minimum  $w$ -multiple antichain covering of  $P(\theta)$ )

**Step 0:** Compute  $(\alpha_i, \beta_i)$  for all  $b_i \in S$ . Set  $S := \{b_i \mid (\alpha_i, \beta_i) \geq (0, 0)\}$  and  $r := 1$ .

**Step 1:** Compute a maximal pareto-optimal set  $S_r$  in  $S$  and remove  $S_r$  from  $S$ .

**Step 2:** If  $S := \emptyset$ , then halt. Otherwise, let  $r := r + 1$  and return to Step 1.  $\square$

Fig. 5 illustrates Algorithm ANTICHAIN-PARTITION. Let  $(S_1, S_2, \dots, S_r)$  be an antichain partition obtained by Algorithm ANTICHAIN-PARTITION. A maximum weighted chain of  $P(\theta)$  can be constructed from  $S_i$ ,  $i = 1, \dots, r$ , as follows. Choose a ball  $b_{i_1}$  from  $S_r$  arbitrarily, and then find a ball  $b_{i_2}$  in  $S_{r-1}$  such that  $(\alpha_{i_2}, \beta_{i_2}) \geq (\alpha_{i_1}, \beta_{i_1})$ . By the maximality of  $S_{r-1}$ , such a  $b_{i_2}$  always exists. We then find a ball  $b_{i_3}$  in  $S_{r-2}$  such that  $(\alpha_{i_3}, \beta_{i_3}) \geq (\alpha_{i_2}, \beta_{i_2})$ . By repeating this argument, we finally have a path  $i_0 = 0, i_1, \dots, i_r$  in  $G(\theta)$ . This path clearly corresponds to a weighted chain  $C$  of  $P(\theta)$ . Since the weight of chain  $C$  is  $r$ , it follows from Theorem 2.6 that  $(S_1, S_2, \dots, S_r)$  corresponding to a minimum  $w$ -multiple antichain covering of  $P(\theta)$  and  $C$  is a maximum weighted chain of  $P(\theta)$ .

Although we skip the details how to compute a maximal pareto-optimal set  $S_i$  in chart  $H(\theta)$ , due to the space limitation, it is not difficult to see that  $S_i$  can be computed (and deleted) in  $O(|S_i| \log n)$  time by using a balanced binary search tree such as AVL- and splay trees to store  $H(\theta)$ . Therefore, we have the following result which improves upon the one in the previous section. <sup>6</sup>

**Theorem 2.7** We can compute in  $O(n \log n)$  time a robot schedule that maximizes the number of the balls collected by a robot moving on a given track-line.  $\square$

Note that the problem contains the sorting problem, since a robot schedule

<sup>6</sup> The same result can be obtained by using  $O(n \log |S_i|)$ -algorithm to compute a maximal pareto-optimal set, such as the one proposed in [13].

decides the ordering of balls caught, i.e., sorts the balls caught in the order of  $t_i$ . Since any (comparison-based) sorting algorithm needs  $\Omega(n \log n)$  time, our algorithm is optimal. We finally remark that our algorithm is based on the fact that each ball crosses at most once, but not the one that balls and robots move on lines, respectively. This means that our method may be applicable in some other situations.

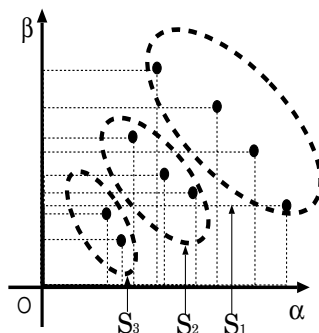


Fig. 5. Algorithm ANTICHAIN-PARTITION

## 2.2 $k$ -robot BCP with a given single track-line

This section considers the case when more than 1 robot can be used. Let us first consider the problem (iii), i.e., minimizing the number of robots to catch all the balls. We show that the problem can be solved in  $O(n \log n)$  time by using the technique shown in the previous section.

Note that the problem (iii) can be reduced to compute a minimum chain covering of the poset  $P(\theta)$ . We solve the problem by applying the algorithm similar to ANTICHAIN-PARTITION to the chart  $H^*(\theta)$  defined below.

Let  $t' = \max_i t_i$ ,  $d' = \min_{i,j: d_i \neq d_j} |d_i - d_j|$ , and  $v' = v + \delta$  where  $0 < \delta < d'/t'$ . Let  $H'(\theta)$  be a 2-dimensional chart with  $t$ - and  $d$ -axes.  $H(\theta)$  has  $n + 1$  points  $(t_i, d_i)$ ,  $i = 0, 1, \dots, n$ , associated with the region  $R'_i$  defined by  $d \leq v'(t - t_i) + d_i$  and  $d \geq -v'(t - t_i) + d_i$ . In other words  $H'$  is obtained from  $H$  by enlarging each region  $R_i$  by a small  $\delta$ . By this modification, no point  $(t_i, d_i)$  is on the boundary of  $R_j$ , if  $(t_i, d_i) \neq (t_j, d_j)$ , and since  $\delta$  is small,  $H'$  still has a good property of  $H$ , i.e., a point  $(t, d)$  is contained in  $R_i$  if and only if a robot starting at state  $(t_i, d_i)$  can reach the state  $(t, d)$ .

Let  $\alpha_i^* = -t_i + d_i/v'$  and  $\beta_i^* = t_i + d_i/v'$ . Let  $H^*(\theta)$  denote the chart of the  $n + 1$  points  $(\alpha_i^*, \beta_i^*)$ ,  $i = 0, 1, \dots, n$ . Similarly to  $H(\theta)$ , the points  $(\alpha_i^*, \beta_i^*)$  ( $i = 1, 2, \dots, n$ ) are identified with balls  $b_i$ . We can see that  $H^*$  is a dual chart of  $H$  in the following sense. Let  $b_i$  and  $b_j$  be balls such that  $(t_i, d_i) \neq (t_j, d_j)$ .

Then  $(\alpha_i, \beta_i)$  and  $(\alpha_j, \beta_j)$  are comparable (i.e.,  $(\alpha_i, \beta_i) \leq (\alpha_j, \beta_j)$  or  $(\alpha_i, \beta_i) \geq (\alpha_j, \beta_j)$ ) if and only if  $(\alpha_i^*, \beta_i^*)$  and  $(\alpha_j^*, \beta_j^*)$  are incomparable.

Since the problem (iii) is to compute a minimum chain covering of the poset  $P(\theta)$  (corresponding to  $H(\theta)$ ), the problem (iii) can be solved by computing a minimum antichain covering of the poset  $P^*(\theta)$  corresponding to  $H^*(\theta)$ . Thus we can use the algorithm similar to ANTICHAIN-PARTITION for the problem. The difference of the algorithms are the following 2 points: Step 0 initializes  $S$  as  $S = \{b_i \mid \alpha_i^* \leq 0, \beta_i^* \geq 0\}$  and Step 1 computes the set of all pareto-optimal points instead of a maximal pareto-optimal set. The second difference follows from the one between a minimum antichain covering and a minimum  $w$ -multiple antichain covering. Similarly to Theorem 2.7, we have the following theorem.

**Theorem 2.8** We can compute in  $O(n \log n)$  time a robot schedule minimizing the number of the robots on a given single track-line that collect all balls.  $\square$

We finally consider the problem (ii) when  $k \geq 2$ . When  $k$  is not enough to collect all balls, we can formulate the problem as a minimum cost network flow problem [1,5] of the following network  $N = (V', A')$  associated with the capacity, cost and demand functions. The network  $N$  can be constructed from  $G(\theta) = (V, A)$  as follows.

$$\begin{aligned} V' &= \{1^+, \dots, n^+\} \cup \{1^-, \dots, n^-\} \cup \{0, n+1\} \\ A' &= \{(i^+, j^-) \mid (i, j) \in A\} \cup \{(i^-, i^+) \mid i = 1, \dots, n\} \\ &\quad \cup \{(0, i^-), (i^+, n+1) \mid i = 1, \dots, n\} \\ \text{demand}(v) &= -k \text{ if } v = 0, \ k \text{ if } v = n+1, \text{ and } 0 \text{ otherwise} \\ \text{cap}(e) &= 1 \text{ for all } e \in A' \\ \text{cost}(e) &= -1 \text{ if } e = (i^+, i^-), \text{ and } 0 \text{ otherwise.} \end{aligned}$$

Since the demand function *demand* is integer, one can compute a minimum cost integral flow. Although the details are omitted due to the space limitation, we can show that integral feasible flows correspond to  $k$ -robot schedules, and moreover, such minimum cost flows correspond to optimal  $k$ -robot schedules. Minimum cost flow problem has been extensively studied [1,5]. Here we make use of *primal-dual algorithm with least-cost augmenting paths* to the network  $N$ . Since  $|V'| = O(n)$ ,  $|A'| = O(n^2)$  and  $k \leq n$ , the problem can be solved in  $O(n^3)$  time.

**Theorem 2.9** We can compute in  $O(n^3)$  time a  $k$ -robot schedule that maximizes the number of the balls collected by  $k$  robots moving on a given track-line.  $\square$

Before concluding this section, we note that, differently from Theorems 2.7 and 2.8, Theorem 2.9 does not use the property (or structure) of  $G(\theta)$ , and hence more efficient algorithms may be possible.

### 3 BCPs with given multiple track-lines

In this section, we address the case that  $k (\geq 2)$  robots move on their own track-line given.

Let  $\theta_j$  be an angle for the track-line  $\ell_j$  which robot  $r_j$  moves on. We first study the problem (i) when  $k = 2$ .

**Theorem 3.1** It can be checked in  $O(n^2)$  time if all balls can be collected by two robots moving on two given track-lines. Furthermore, if so, such a robot schedule can be computed in  $O(n^2)$  time.  $\square$

**Proof.** We reduce the problem to the problem 2-SAT whose size is  $O(n^2)$ . Since 2-SAT is solvable in linear time [6], our problem is solvable in  $O(n^2)$  time.

We define a Boolean variable  $u_i$  for each ball  $b_i$ . Here  $u_i = 1$  means that  $b_i$  is caught by robot  $r_1$  and  $u_i = 0$  means that  $b_i$  is caught by robot  $r_2$ . A 2-CNF  $\psi$  contains the following 4 types of clauses.

1.  $\bar{u}_i$ , if  $G(\theta_1)$  contains no arc  $(0, i)$ .
2.  $(\bar{u}_i \vee \bar{u}_{i'})$ , if  $G(\theta_1)$  contains neither arc  $(i, i')$  nor arc  $(i', i)$ .
3.  $u_i$ , if  $G(\theta_2)$  contains no arc  $(0, i)$ .
4.  $(u_i \vee u_{i'})$ , if  $G(\theta_2)$  contains neither arc  $(i, i')$  nor arc  $(i', i)$ .

It is easy to see that the size of  $\psi$  is  $O(n^2)$ . Note that 3 and 4 are dual of 1 and 2, respectively. By the definitions of  $G(\theta_1)$  and  $G(\theta_2)$ , if all balls can be collected by two robots,  $\psi$  is satisfiable. On the other hand, if  $\psi$  is satisfiable, we shall below show that all balls can be collected. Let  $u^*$  be an assignment such that  $\psi(u^*) = 1$ , and let  $S_1 = \{i \mid u_i^* = 1\}$  and  $S_2 = \{i \mid u_i^* = 0\}$ . For any pair of  $i$  and  $i'$  in  $S_1$ ,  $G(\theta_1)$  contains arc  $(i, i')$  or  $(i', i)$ . It follows from the transitivity of  $G(\theta_1)$  that it must contain a directed path through all the nodes in  $S_1$ . This implies that  $S_1$  can be caught by robot  $r_1$ . Similarly,  $S_2$  can be caught by robot  $r_2$ . Since  $S_1 \cup S_2 = \{1, 2, \dots, n\}$ , two robots can collect all balls. This completes the proof.  $\square$

Unfortunately, the problem becomes intractable if we generalize the problem slightly.

**Theorem 3.2** (i) It is NP-hard to maximize the number of the balls collected by 2 robots with 2 given track-lines.

(ii) It is NP-complete to decide if all balls can be collected by  $k$  robots with given track-lines, if  $k \geq 3$ .  $\square$

The next corollary follows from Theorem 3.2 (ii).

**Corollary 3.3** It is NP-hard to minimize the number of the robots with given track-lines that collect all balls.  $\square$

## 4 BCPs with arbitrary track-lines

In Sections 2 and 3, we consider the BCPs under the assumption that track-lines  $\ell_j$  are given in advance. In this section, we consider the case in which a track-line of each robot can be chosen as a part of the robot scheduling.

### 4.1 Selecting track-lines

This section shows that the number of the track-lines that have to be considered for the BCPs is polynomial, although there exist infinitely many lines through the origin.

Let  $\theta$  denote the angle of a line  $\ell$  through the origin. For any ordered pair of  $i$  and  $j$  in  $V (= \{0, 1, \dots, n\})$ , let

$$I_{ij} = \{\theta \in [0, \pi) \mid G(\theta) \text{ contains arc } (i, j)\}.$$

Moreover, let  $I_{ij} = \bigcup_p I_{ij}^p$ , with interval  $I_{ij}^p$  in  $[0, \pi)$ . Note that  $I_{ij}^p$  might not be closed.

**Lemma 4.1** The number of the intervals  $I_{ij}^p$  representing  $I_{ij}$  for  $i, j \in V$  is  $O(1)$ , and the interval representation of  $I_{ij}$  can be computed in  $O(1)$  time.  $\square$

The proof is omitted, since it is rather complicated. The lemma immediately implies the following lemma, since we have  $n^2$  ordered pairs  $(i, j)$ .

**Lemma 4.2** There exists  $O(n^2)$  different directed graphs  $G(\theta)$ ,  $\theta \in [0, \pi)$ .  $\square$

One might see that  $O(n^2)$  is overestimated, since  $G(\theta)$  contains many transitive arcs, in general. However, we can construct a problem instance having  $\Omega(n^2)$  directed graphs  $G(\theta)$ . By combining Lemma 4.2 with the results in the previous sections, we have the following results.

**Theorem 4.3** Assume that a track-line of each robot can be decided as a part of the robot scheduling. Then

(i) We can compute in  $O(n^3 \log n)$  time a robot schedule that maximizes the number of the balls collected by a robot.

(ii) We can compute in  $O(n^3 \log n)$  time a robot schedule minimizing the number of the robots on a single track-line that collect all balls.

(iii) We can compute in  $O(n^5)$  time a  $k$ -robot schedule that maximizes the number of the balls collected by  $k$  robots moving on a single track-line.

(iv) It can be checked in  $O(n^6)$  time if all balls can be collected by two robots moving on two track-lines. Furthermore, if so, such a robot schedule can be computed in  $O(n^6)$  time.  $\square$

Theorem 4.3 (i), (ii), (iii), and (iv) follow from Theorems 2.7, 2.8, 2.9 and 3.1, respectively.

#### 4.2 Collecting all balls by a single robot

Theorem 4.3 (i) implies that we can decide whether all balls can be collected by a single robot in  $O(n^3 \log n)$  time. This section shows that a faster algorithm is possible for the problem.

For the angle  $\theta$  of a line  $\ell$ , let  $E(\theta) = \{(i, j) \mid i < j, G(\theta) \text{ contains arc } (i, j) \text{ or } (j, i)\}$ .

**Lemma 4.4** All balls can be collected by a robot on a line  $\ell$  if and only if  $|E(\theta)| = n(n+1)/2$ .  $\square$

This lemma implies that it is not necessary to check if  $G(\theta)$  has a directed path from 0 whose length is  $n$ , but is sufficient to check the size of  $E(\theta)$ .

Here we only present an outline of the algorithm, due to the space limitation. Recall that  $I_{ij}$  can be represented by the union of  $O(1)$  intervals. We first compute such representations of all  $I_{ij}$ , which requires  $O(n^2)$  time. Let  $\mathcal{I}$  denotes the family of all such intervals  $I = \langle L, R \rangle$ . Here  $\langle$  is either  $[$  or  $($ , and  $\rangle$  is either  $]$  or  $)$ . Then we sort all the left and right elements  $L$  and  $R$  in  $\mathcal{I}$ , which is possible in  $O(n^2 \log n)$  time. Let  $J_1, J_2, \dots, J_q$  be such an ordering. We finally check if  $|E(\theta)| = n(n+1)/2$  for all  $\theta = 0, J_1 + \epsilon, J_2 + \epsilon, \dots, J_q + \epsilon$  in this order, where  $\epsilon$  is sufficiently small positive.  $\epsilon$  is used for open intervals. It is not difficult to see that these angles are sufficient to check. Since this can be done in  $O(n^2)$  time, we have an  $O(n^2 \log n)$  time algorithm.

**Theorem 4.5** It can be checked in  $O(n^2 \log n)$  time if there exists a track-line  $\ell$  such that a robot can collect all balls by moving on  $\ell$ . Furthermore, if so, such a robot schedule (including  $\ell$ ) can be computed in  $O(n^2 \log n)$  time.  $\square$

#### 4.3 The NP-hardness cases

Similarly to Theorem 3.2, we have the negative results even if the track-lines can be chosen.

**Theorem 4.6** Even if a track-line of each robot can be chosen as a part of the robot scheduling, the following statements holds.

(i) It is NP-hard to maximize the number of the balls collected by 2 robots with 2 track-lines.

(ii) It is NP-complete to decide if all balls can be collected by  $k$  robots with track-lines, if  $k \geq 3$ .  $\square$

## 5 Conclusion and Discussions

We have studied the problem of collecting balls moving in the Euclidean plane by robots moving on straight lines through the origin. From the viewpoint of computational complexity, we have investigated the following 3 problems in various situations: (i) deciding if  $k$  robots can collect all  $n$  balls, (ii) maximizing the number of the balls collected by  $k$  robots, and (iii) minimizing the number of the robots to collect all  $n$  balls. The results are summarized in Table 1 in Section 1.

There are many possible extensions of the problems. For example, we can relax the track shapes of robots and/or balls. Unfortunately, it is shown that the problems become intractable, if robots and/or balls move on general tracks.

## Acknowledgements

The authors would like to thank Prof. Tiko Kameda and Prof. Ichiro Suzuki for their helpful comments. This work was partially supported by the Scientific Grant-in-Aid by the Ministry of Education, Science, Sports and Culture of Japan.

## References

- [1] Ahuja, R. K., T. L. Magnanti and J. B. Orlin, *Network Flows : Theory, Algorithms and Applications*, Prentice-Hall, 1993.
- [2] Atallah, M. and S. Kosaraju, Efficient solutions to some transportation problems with applications to minimizing robot arm travel, *SIAM Journal on Computing*, 17, pp.849–869, 1988.
- [3] Bauernöppel, F., E. Kranakis, D. Krizanc, A. Maheshwari, J.-R. Sack, J. Urrutia, Planer stage graphs: Characterizations and applications, *Theoretical Computer Science*, 175, pp. 239–255, 1997.
- [4] Chalasani, P., R. Motwani and A. Rao, Approximation Algorithms for Robot Grasp and Delivery, *2nd International Workshop on Algorithmic Foundations of Robotics (WAFR)*, pp.347–362, 1996.



- [5] Cook, W. J., W. H. Cunningham, W. R. Pulleyblank and A. Schrijver, *Combinatorial Optimization*, John Wiley & Sons, 1998.
- [6] Evan, S., A. Itai and A. Shamir, On the Complexity of Timetabling and Multicommodity Flow Problems, *SIAM Journal of Computing*, 5(4), pp.691–703, 1976.
- [7] Garey, M. R. and D. S. Johnson, *Computers and Intractability*, Freeman, New York, 1979.
- [8] Hammar, M. and B. J. Nilsson, Approximation results for kinetic variants of TSP, *26th International Colloquium, Automata, Languages and Programming, ICALP'99*, LNCS 1644, pp.392–401, 1999.
- [9] Helvig, C. S., G. Robins, and A. Zelikovsky, Moving Target TSP and Related Problems, *6th Annual European Symposium, Algorithms - ESA '98*, LNCS 1461, pp.453–464, 1998.
- [10] Karuno, Y., H. Nagamochi and T. Ibaraki, Better Approximation Ratios for the Single-Vehicle Scheduling Problems on Line-Shaped Networks, *Networks*, 39(4), pp. 203–209, 2002.
- [11] Knuth, D. E., *Fundamental Algorithms, Third Edition*, Addison-Wesley, 1997.
- [12] Lovász, L. and M.D. Plummer, *Matching Theory*, Annals of Discrete Mathematics, 29, North-Holland, 1986.
- [13] Nielsen, F., Output-sensitive peeling of convex and maximal layers, *Information Processing Letters*, 59, pp. 255 – 259, 1996.
- [14] Psaraftis, H. N., M. M. Solomon, T. L. Magnanti, and T. Kim, Routing and Scheduling on a Shoreline with Release Times, *Management Science*, 36(2), pp. 212–223, 1990.
- [15] G. Rote, The N-line traveling salesman problem, *Networks*, 22, pp.91–108, 1992.
- [16] Tsitsiklis, J. N., Special cases of traveling salesman and repairman problems with time windows, *Networks* 22, pp.263–282, 1992.