RC24328 (W0708-033) August 8, 2007
Computer Science

IBM Research Report

Algorithms for Capacitated Rectangle Stabbing and Lot Sizing with Joint Set-Up Costs

Guy Even[1], Retsef Levi[2], Dror Rawitz[3], Baruch Schieber[4], Shimon (Moni) Shahar[1], Maxim Sviridenko[4]

[1]School of Electrical Engineering
Tel-Aviv University
Tel-Aviv, Israel

[2]Sloan School of Management
MIT
Cambridge, MA 02142

[3]Caesarea Rothschild Institute
University of Haifa
Haifa 31905, Israel

[4]IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Algorithms for Capacitated Rectangle Stabbing and Lot Sizing with Joint Set-Up Costs

Guy Even* Retsef Levi† Dror Rawitz‡ Baruch Schieber§
Shimon (Moni) Shahar* Maxim Sviridenko§

August 1, 2007

Abstract

In the rectangle stabbing problem we are given a set of axis parallel rectangles and a set of horizontal and vertical lines, and our goal is to find a minimum size subset of lines that intersect all the rectangles. In this paper we study the capacitated version of this problem in which the input includes an integral capacity for each line. The capacity of a line bounds the number of rectangles that the line can cover. We consider two versions of this problem. In the first, one is allowed to use only a single copy of each line (*hard capacities*), and in the second, one is allowed to use multiple copies of every line, but the multiplicities are counted in the size (or weight) of the solution (*soft capacities*).

We present an exact polynomial-time algorithm for the weighted one dimensional case with hard capacities that can be extended to the one dimensional weighted case with soft capacities. This algorithm is also extended to solve a certain capacitated multi-item *lot sizing* inventory problem with joint set-up costs. For the case of d-dimensional rectangle stabbing with soft capacities, we present a $3d$-approximation algorithm for the unweighted case. For d-dimensional rectangle stabbing problem with hard capacities, we present a bi-criteria algorithm that computes $4d$-approximate solutions that use at most two copies of every line. Finally, we present hardness results for rectangle stabbing when the dimension is part of the input and for a two-dimensional weighted version with hard capacities.

Keywords: Approximation Algorithms; Capacitated Covering; Rectangle Stabbing; Lot Sizing.

*School of Electrical Engineering, Tel-Aviv University, Tel-Aviv, Israel. {guy, moni}@eng.tau.ac.il.

†Sloan School of Management, MIT, Cambridge, MA 02142. retsef@mit.edu. Part of this work was done while this author was a postdoctoral fellow at IBM T.J. Watson Research Center.

‡Caesarea Rothschild Institute, University of Haifa, Haifa 31905, Israel. rawitz@cri.haifa.ac.il.

§IBM T.J. Watson Research Center, Yorktown Heights, NY 10598. {sbar, sviri}@us.ibm.com.

1 Introduction

Understanding the combinatorial and algorithmic nature of capacitated covering problems is still an open problem. Only a few capacitated problems were studied including the general case of Set Cover [19] and the restricted case of Vertex Cover [5, 12]. Capacity constraints appear naturally in many applications, for example, bounded number of clients an antenna can serve. In this paper we consider a capacitated version of a covering problem, called rectangle stabbing. The geometric nature of the problem is used to obtain exact algorithms and constant approximation ratios for some versions of the problem.

1.1 The problems

The *rectangle stabbing* problem (RS) is a covering problem. Its uncapacitated version is defined as follows. The input is a finite set $\mathcal{U}$ of axis parallel rectangles and a finite set $\mathcal{S}$ of horizontal and vertical lines. A *cover* is a subset of $\mathcal{S}$ that intersects every rectangle in $\mathcal{U}$ at least once. The goal is to find a cover of minimum size. We denote the set of rectangles that a line $S \in \mathcal{S}$ intersects by $\mathcal{U}(S)$. Using this notation, an RS instance is simply a Set Cover instance in which the goal is to find a collection of subsets of the form $\mathcal{U}(S)$, the union of which equals $\mathcal{U}$. Without loss of generality, one may assume that the RS instance is discrete in the following sense [10]: rectangle corners have integral coordinates and lines intersect the axes at integral points.

In the one-dimensional version, the set $\mathcal{U}$ consists of horizontal intervals and the set $\mathcal{S}$ consists of points. This is the well known clique cover problem in interval graphs that is solvable in polynomial time [11]. The RS problem can be extended to d dimensions (d-RS). For $d \geq 3$, the set $\mathcal{U}$ consists of axis parallel d-dimensional rectangles (i.e., "boxes") and the set $\mathcal{S}$ consists of hyperplanes that are orthogonal to one of the d axes (i.e., "walls"). In the sequel we stick to the two-dimensional terminology, that is, we refer to $\mathcal{U}$ as a set of rectangles and to $\mathcal{S}$ as a set of lines.

Rectangle stabbing is a special case of the problem of hitting two-dimensional objects by lines (see Section 1.2 for a brief overview). Applications related to numeric computation and image processing motivated the investigation of the rectangle stabbing problem by Gaur et al. [10]. We introduce capacity constraints that model the property that every covering object has limited resources that bound the number of elements it can cover. A limited covering ability of a covering object can occur in situations where covering each element consumes time or power. We now define the capacitated version of RS.

In the *capacitated d-dimensional rectangle stabbing* problem the input includes an integral capacity $c(S)$ for every line $S \in \mathcal{S}$. The capacity $c(S)$ bounds the number of rectangles that S can cover. This means that in the capacitated case one has to specify which line covers each rectangle. The assignment of rectangles to lines may not assign more than $c(S)$ rectangles to a line S. We discuss two variants of capacitated d-dimensional rectangle stabbing called covering with hard capacities (HARD-d-RS) and covering with soft capacities (SOFT-d-RS).

A cover in SOFT-d-RS is formally defined as follows. The input consists of a set $\mathcal{U}$ of d-dimensional axis-parallel rectangles and a set $\mathcal{S}$ of lines (i.e., hyperplanes) that are orthogonal to one of the d axis. Each line $S \in \mathcal{S}$ is given a nonnegative integral capacity $c(S)$. An *assignment* is a function $A : \mathcal{S} \to 2^{\mathcal{U}}$, where $A(S) \subseteq \mathcal{U}(S)$, for every S. A rectangle u is *covered* by a line S if $u \in A(S)$. An assignment A is a *cover* if every rectangle is covered by some line, i.e., $\bigcup_{S \in \mathcal{S}} A(S) = \mathcal{U}$. The *multiplicity* (or number of copies) of a line $S \in \mathcal{S}$ in an assignment A equals $\lceil |A(S)|/c(S) \rceil$. We denote the multiplicity of S in A by $\alpha(A, S)$. The *size* of a cover A is the sum $\sum_{S \in \mathcal{S}} \alpha(A, S)$. The goal is to find a cover of minimum size.

Given the multiplicities of every line in a cover A, one can compute a cover with the same

multiplicities by solving a flow problem. We therefore often refer to a cover simply as a multi-set of lines. The *support* of an assignment A is the set of lines $\{S \in \mathcal{S} : A(S) \neq \emptyset\}$. Note that the support is a set and not a multi-set. We denote the support of A by $\sigma(A)$.

In HARD-d-RS, a line may appear at most once in a cover. Hence, in this case, a cover is an assignment A for which $|A(S)| \leq c(S)$, (or $\alpha(A,S) \in \{0,1\}$) for every $S \in \mathcal{S}$. In this setting, we refer to a cover as the set of lines it contains (i.e., its support). Note that SOFT-d-RS is a special case of HARD-d-RS, since given a SOFT-d-RS instance one can always transform it into a HARD-d-RS instance by duplicating each line $|\mathcal{U}|$ times.

All the problems mentioned above have weighted versions, in which we are given a weight function w defined on the lines. In this case the weight of a cover A is $w(S) = \sum_S \alpha(A,S) \cdot w(S)$, and the goal is to find a cover of minimum weight.

We also consider a variant of a *multi-item lot sizing inventory problem* (MILS) that generalizes HARD-1-RS. (The reduction is shown in Section 3.) In this multi-item lot sizing problem there is a sequence of unit size *orders* denoted by $O_1, \ldots, O_n$ that need to be satisfied over a planning horizon of T discrete periods, indexed $t = 1, \ldots, T$. We henceforth refer to orders as *requests* to avoid confusion with total ordering. Each request O_i has a due date d_i, which means that it must be manufactured at some time period $s \leq d_i$.

Production takes place in mixed batches of bounded capacity. Specifically, each time period $t = 1, \ldots, T$ is associated with a *capacity* $c(t)$ and weight $w(t)$. The capacity $c(t)$ bounds the number of requests that can be manufactured at time period t, and $w(t)$ is a fixed manufacturing cost at time period t for any positive number of requests up to $c(t)$.

In addition, there are costs for maintaining inventory, traditionally called *holding costs*. For example, if the request O_i is manufactured at some period $s \leq d_i$, there are holding costs incurred by carrying this request in inventory from period s to period d_i. Let $H_i(s)$ denote the holding cost incurred by request O_i given that it is manufactured at time period $s \leq d_i$. We assume that, for each $i = 1, \ldots, n$, the function $H_i(s)$ is non-negative and non-increasing in $s \in (0, d_i]$, i.e., shortening the holding time always results in lower holding cost. (We note that one can incorporate a request-specific production cost into the function $H_i(s)$.) We also assume that the requests are indexed by increasing order of *importance*. (Without loss of generality, we assume that no two requests have the same importance.) Suppose that $i < j$. Then, $H_i(s_1) + H_j(s_2) \leq H_i(s_2) + H_j(s_1)$, for $s_1 < s_2 \leq \min\{d_i, d_j\}$. In words, shortening the holding time of the more important requests at the expense of extending the holding time of the less important requests never increases the overall holding costs.

Assume that it is possible to satisfy all requests; namely, $\sum_{t=0}^{d_i} c(t) \geq i$, for $i = 1, \ldots, n$. The goal is to find a feasible manufacturing schedule with the least cost.

1.2 Previous results

Since 1-RS is equivalent to clique cover in interval graphs, it can be solved in linear time [11]. Hassin and Megiddo [13] studied the problem of hitting two-dimensional objects by lines. They proved that d-RS is NP-hard, for $d \geq 2$. They also focused on lines of restricted line slopes (e.g., horizontal and vertical lines) and objects that are horizontal segments (i.e., intersected by a unique horizontal line).Gaur et al. [10] presented a d-approximation algorithm for d-RS that uses linear programming to reduce d dimensions to one dimension. Kovaleva and Spieksma [14] considered weighted two-dimensional RS where each rectangle is given a demand specifying a lower bound on the number of times it must be stabbed and where a solution may contain multiple copies of the same line. Based on [10], they presented a $(q+1)/q$-approximation algorithm for the case that the minimum demand is q. For the special case of RS in which each rectangle is stabbed by exactly one

horizontal line, they presented an $e/(e-1)$-approximation algorithm.

Capacitated covering problems (even with weights) date back to Wolsey [19] (see also [4, 5]). Wolsey presented a greedy algorithm for weighted Set Cover with hard capacities that achieves a logarithmic approximation ratio. Guha et al. [12] presented a 2-approximation primal-dual algorithm for the weighted Vertex Cover problem with soft capacities. Chuzhoy and Naor [5] presented a 3-approximation algorithm for Vertex Cover with hard capacities (without weights) which is based on randomized rounding with alterations. They also proved that the weighted version of this problem is as hard to approximate as set cover. Gandhi et al. [9] improved the approximation ratio for capacitated Vertex Cover to 2.

Anily and Tzur [1] have considered a closely related inventory model. In particular, in their model there are n requests, not necessarily unit size, and the capacities are uniform (i.e., $c(t) = C$, for each $t = 1, \ldots, T$). Their cost structure is a special case of ours, where $w(t) = w$, for each $t = 1, \ldots, T$, and the holding cost of any request is linear in the number of periods it is kept in inventory (i.e., the function $H_i(s)$ is linear). They presented an optimization dynamic-programming-based algorithm whose running time is $O(nT^{n+5})$ for this special case, where again n denotes the number of requests and T the number of periods. Later, Anily, Tzur and Wolsey [2] proposed a polynomial size linear program that solves a more general model with uniform capacities and time-dependent cost parameters. In particular, there is a per-unit holding cost for each time period and each request, and the overall holding costs incurred by producing the request O_i at time period s is equal to the sum of the corresponding unit holding costs over the interval $[s, d_i)$ (i.e., the function $H_i(s)$ is piecewise linear). In addition, there is a notion of relative *importance* of the different requests. Specifically, for each time period, the per-unit holding cost of a request is larger than the per-unit holding cost of a less important request. Compared to the model considered by Anily, Tzur and Wolsey [2], in MILS there are only unit size (or polynomially bounded size) requests, but a more general holding cost structure and time-dependent capacities. However, it is well known that if we allow arbitrary request sizes, MILS becomes NP-hard. (There is a reduction from knapsack [8].) Furthermore, recently Levi, Lodi and Sviridenko [16] have shown that without the relative importance property of the requests, the model considered by Anily, Tzur and Wolsey is strongly NP-hard even for unit size requests. (Clearly, the same applies to MILS.)

1.3 Our results

We present a dynamic programming algorithm for weighted HARD-1-RS which implies also an exact algorithm for weighted SOFT-1-RS. The running time of the algorithm is $O((|\mathcal{U}|^2|\mathcal{S}|^2)(|\mathcal{U}| + |\mathcal{S}|))$. This algorithm extends to solve also the multi-item lot sizing problem in time $O((n^2T^2)(n+T))$. Our dynamic programming algorithm is motivated by a paper by Baptiste [3]. We note that a 2-approximation algorithm for weighted SOFT-1-RS whose running time is $O(|\mathcal{U}|^2 \cdot |\mathcal{S}|)$ was presented in [6] (see also [18]).

We present $3d$-approximation algorithm for SOFT-d-RS, where d is arbitrary. This algorithm solves an LP relaxation of the problem, and rounds it using the geometrical structure of the problem. For the case of hard capacities we show that the same technique can be used to obtain a bi-criteria algorithm for HARD-d-RS that computes solutions that are $4d$-approximate and use at most two copies of each line. We note that these techniques were extended in [18] to obtain an 8-approximation algorithm for HARD-1-RS. It follows that the integrality gap of the natural LP for HARD-1-RS is bounded by 8.

Finally, we present two hardness results. The first result mimics the hardness result given in [5], to show that weighted HARD-2-RS is Set Cover hard, even if all weights are in $\{0, 1\}$. The second hardness result proves that it is NP-hard to approximate d-RS with a ratio of $c \cdot \log d$, for some

constant c. Note that the dimension d is considered here to be part of the input.

2 One dimensional rectangle stabbing with hard capacities

In this section we present an exact algorithm for weighted HARD-1-RS. Since SOFT-1-RS is a special case of HARD-1-RS, this also implies an exact algorithm for weighted SOFT-1-RS. We mention how the algorithm can be slightly modified to apply to weighted SOFT-1-RS.

In the one-dimensional case of the capacitated rectangle stabbing problems rectangles are simply intervals that we draw as horizontal intervals. To facilitate the task of drawing overlapping intervals, we separate intervals by drawing them at different heights. Hyperplanes in the one dimensional case are simply points. Since intervals are drawn as horizontal intervals with different heights, we refer to the hyperplanes as vertical lines instead of points. To summarize, the input in HARD-1-RS consists of a set $\mathcal{U}$ of horizontal intervals and a set $\mathcal{S}$ of vertical lines with capacities $c(S)$.

We use the following notation. Given an interval u, we denote the coordinates of its endpoints by $\ell(u) < r(u)$. We assume, without loss of generality, that the coordinates are integers between 1 and $\max\{2|\mathcal{U}|, |\mathcal{S}|\}$, and that there is no more than one endpoint and no more than one vertical line at each coordinate. Order the intervals by their left endpoint and denote this order by $u_1, u_2, \ldots, u_{|\mathcal{U}|}$ (i.e., $\ell(u_i) < \ell(u_{i+1})$). For a vertical line $S \in \mathcal{S}$, let $x(S)$ denote the x-coordinate of S. Inversely, for a coordinate x let $S(x)$ be the vertical line whose x-coordinate is x, if such exists.

Without loss of generality, we look for an optimal solution that has the following *leftmost interval first* property (if time is reversed, this is the well known earliest due-date scheduling policy): For any line $S \in \mathcal{S}$ that is in the optimal solution, if S covers an interval $u \in \mathcal{U}$, then all lines $S' \in \mathcal{S}$ with $\ell(u) \leq x(S') < x(S)$ that are also in the solution are used to their full capacity and for all intervals $u' \in \mathcal{U}$ covered by these lines either $\ell(u') < \ell(u)$ or $r(u') < x(S)$. In words, interval u cannot be covered by any line that is to the left of S, and cannot be swapped with any interval u' with $\ell(u') > \ell(u)$ that is covered by such a line.

Notice that given an optimal assignment A that does not satisfy this property, one can perform a series of corrections until the property holds. Namely, as long as the assignment does not satisfy the property, each interval u is moved to the leftmost vacant line that can cover it, and any two intervals u and u' covered by lines S and S' respectively, are swapped if $\ell(u) < \ell(u') \leq x(S') < x(S) \leq r(u')$.

Satisfying the leftmost interval first property implies the following observation.

Observation 1. *Let A be an optimal assignment that satisfies the leftmost interval first property. For any range $[x_1, x_2]$, let u be the interval with the minimum $\ell(u)$ among the intervals covered by lines in this range. If u is covered by line S, where $x = x(S)$, then the right endpoint of all intervals covered by lines in the range $[x_1, x-1]$ are to the left of x.*

Proof. Assume that there exists an interval u that is covered by a line in the range $[x_1, x-1]$ and $r(u) \geq x$. Since u and u' can be swapped, it follows A that does not satisfy the leftmost interval first property. ■

To define the dynamic programming we need the following notation. For interval index $i \in [1..n]$ and two coordinates $x_1 \leq x_2$, such that $r(u_i) \in [x_1, x_2]$, let $\mathcal{U}(i, x_1, x_2) = \{u_j \mid j \geq i \wedge r(u_j) \in [x_1, x_2]\}$. That is, $\mathcal{U}(i, x_1, x_2)$ is the set of all intervals whose left endpoint is at or to the right of $\ell(u_i)$ and whose right endpoint falls within the range $[x_1, x_2]$.

The dynamic programming table Π of size $O(|\mathcal{S}|^2 \cdot |\mathcal{U}|^2)$ is defined as follows. The entry $\Pi(i, x_1, x_2, k)$, where $i \in [1..n]$ is an interval index, x_1 and x_2 are x-coordinate of lines in $\mathcal{S}$, and $r(u_i) \in [x_1, x_2]$, contains the minimum weight of a cover of the intervals in $\mathcal{U}(i, x_1, x_2)$ by lines

whose x-coordinate is in the range $[x_1, x_2]$ with the additional constraint that the line $S = S(x_1)$ covers no more than $k \leq c(S)$ intervals, and if $k < c(S)$ then the weight of S is assumed to be zero.

Let S_L and S_R be the lines with the leftmost and rightmost x-coordinate. Then the optimal cover is given by $\Pi(1, x(S_L), x(S_R), c(S_L))$.

The base case $\Pi(i, x_1, x_1, k)$ is simple since only a single line $S(x_1)$ is considered. Specifically, $\Pi(i, x_1, x_1, k) = \infty$, if $k < |\mathcal{U}(i, x_1, x_1)|$, since then the subproblem is infeasible. Else, $\Pi(i, x_1, x_1, k) = w(S)$, if $k = c(S)$ and 0 otherwise. Also, $\Pi(i, x_1, x_2, 0)$ is $\Pi(i, x', x_2, c(x'))$ if $\mathcal{U}(i, x_1, x_2) = \mathcal{U}(i, x', x_2)$, where x' is the x-coordinate of the next line to the right of $S(x_1)$, and ∞ otherwise since then the subproblem is infeasible.

Below, we show how to compute an entry $\Pi(i, x_1, x_2, k)$, for $x_1 < x_2$ and $k > 0$ in polynomial time given all entries $\Pi(i', x'_1, x'_2, k')$ with $i' > i$. Since the size of the table is polynomial this implies a polynomial time algorithm. The computation is based on the Observation 1. To compute $\Pi(i, x_1, x_2, k)$ we enumerate over all possible lines that can cover the interval u_i. Consider such a line S. The coordinate $x(S)$ partitions the problem into two subproblems: a *left instance* that contains all intervals in $\mathcal{U}(i, x_1, x_2)$ whose right endpoint is to the left of $x(S)$ which must be covered by lines to the left of S, and a *right instance* that contains the rest of the intervals (excluding u_i). By Observation 1 these intervals are covered by either S or lines to its right. See Figure 1. This implies the following relation.

$$\Pi(i, x_1, x_2, k) = \min_{\substack{x \in [x_1, \min\{r(u_i), x_2\}] \\ \wedge x \in \{x(S) \mid S \in \mathcal{S}\}}} \left\{ \Pi(i', x_1, x', k) + \Pi(i'', x, x_2, k'') + w'(S(x)) \right\} \tag{1}$$

where:

- $i' = \min\{j : (j > i) \wedge (x_1 \leq r(u_j) < x)\}$
- $i'' = \min\{j : (j > i) \wedge (x \leq r(u_j) \leq x_2)\}$
- $x' = \max\{y : (x_1 \leq y < x) \wedge (y = x(S)) \text{ for some } S \in \mathcal{S}\}$
- $k'' = \begin{cases} c(S(x)) - 1 & x > x_1, \\ k - 1 & x = x_1 \end{cases}$
- $w'(S(x)) = \begin{cases} w(S(x)) & x > x_1, \\ w(S(x)) & x = x_1 \wedge k = c(S(x_1)), \\ 0 & \text{otherwise.} \end{cases}$

The idea in Equation 1 is that if u_i is covered by S in the interval $[x_1, x_2]$, then the subproblem $\mathcal{U}(i, x_1, x_2)$ is partitioned into a left instance and a right instance. The left instance is $\mathcal{U}(i', x_1, x')$, where i' is the interval with left-most left endpoint among the intervals whose right endpoint is before $x(S)$, and x' is the coordinate of the first line to the right of x_1. We assume that $\Pi(i', x_1, x', k)$ is the optimum of the left instance. The right instance is $\mathcal{U}(i'', x(S), x_2)$, where i'' is the interval with left-most left endpoint among the intervals whose right endpoint is not to the left of $x(S)$, and k'' is the residual capacity of the line whose coordinate is $x(S)$. We assume that $\Pi(i'', x, x_2, k'')$ is the optimum of the right instance. The cost of covering u_i by S is denoted by $W'(S(x))$

Note that computing an entry takes linear time and thus the overall time complexity is $O(|\mathcal{U}|^2|\mathcal{S}|^2(|\mathcal{U}| + |\mathcal{S}|))$. The computation of $\Pi(i, x_1, x_2, k)$ can be modified to compute a corresponding optimal solution.

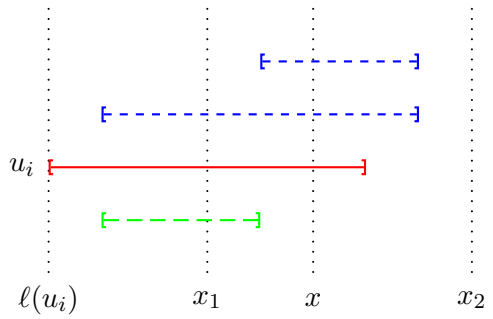


Figure 1: Example of a computing $\Pi(i, x_1, x_2, k)$ in the dynamic programming table. The interval at the bottom belongs to $\mathcal{U}(i', x_1, x')$, while the two intervals at the top belong to $\mathcal{U}(i'', x, x_2)$.

We now turn to SOFT-1-RS. One simple way to solve SOFT-1-RS is to make $\left\lceil \frac{\mathcal{U}(S)}{c(S)} \right\rceil$ copies of each lines S, and execute the algorithm for HARD-1-RS. This straightforward reduction may increase the running time. Alternatively, one may modify the algorithm to handle soft capacities. First, the computation of the base case should take into account the possibility of using multiple copies. Second, change the way k'' is computed. Namely, to solve SOFT-1-RS k'' is set as follows:

$$k'' = \begin{cases} c(S(x)) - 1 & x > x_1, \\ k - 1 & x = x_1 \wedge k > 1, \\ c(S(x_1)) & x = x_1 \wedge k = 1. \end{cases}$$

Note that each time k goes to zero we set it back to the original capacity indicating the use of a new copy of the line $S(x)$.

3 Multi-Item Lot Sizing

In this section we show how to extend the dynamic programming algorithm for the weighted HARD-1-RS to solve the multi-item lot sizing problem.

Recall that an MILS instance consists of n requests $O_1, \ldots, O_n$ each for a single item. Each request O_i has a due date d_i. Without loss of generality, we assume that due dates are distinct. To satisfy a request O_i the respective item needs to be manufactured at some time period $s \leq d_i$, and in this case we are required to pay a holding cost of $H_i(s)$. We also assume that there is a complete *importance* relation on the requests. Specifically, assume that requests are indexed by increasing order of importance. Then if $i < j$, we have $H_i(s_1) + H_j(s_2) \leq H_i(s_2) + H_j(s_1)$, for $s_1 < s_2 \leq \min\{d_i, d_j\}$. Each time period t is associated with a *capacity* $c(t)$ and weight $w(t)$. The capacity $c(t)$ bounds the number of requests that can be manufactured at time t, and $w(t)$ is a fixed manufacturing cost at time period t for any positive number of requests up to $c(t)$. Our goal in MILS is to find a feasible manufacturing schedule with the least cost.

We say that a schedule *respects* priorities if, for every two requests O_i and O_j where $i < j$, whenever both requests are manufactured no later than $\min\{d_i, d_j\}$, then O_i is manufactured no later than job O_j.

We first show that this problem is a general case of HARD-1-RS. Given an instance of HARD-1-RS we show how to translate it to a lot sizing problem instance. Each interval $u_i \in \mathcal{U}$ in the HARD-1-RS instance is associated with a request O_i in the respective lot sizing instance. The due date of the request O_i is the right endpoint of u_i. The capacity at time t is $c(S)$ if there is a point S located at t in the HARD-1-RS instance, and 0 otherwise. The holding cost is defined as

$$H_i(s) = \begin{cases} 0 & s \geq \ell(u_i), \\ \infty & \text{otherwise.} \end{cases}$$

It follows that the importance of the requests in the lot sizing instance is determined by the left endpoints of the intervals in the HARD-1-RS instance.

Any finite weight priority respecting solution to the MILS instance corresponds to a solution to the respective HARD-1-RS instance that has the *leftmost interval first* property with the same weight (cost). Similarly, any solution to the HARD-1-RS instance with the leftmost interval first property corresponds to a priority respecting solution to the respective lot sizing problem instance with the same cost (weight).

The dynamic programming for the lot sizing problem is based on the following observation which is similar to the corresponding observation for HARD-1-RS solution with the the leftmost interval first property.

Observation 2. *For any time interval $[t_1, t_2]$, let O be the request with the least importance among the requests manufactured in this time interval. Suppose that request O is manufactured at time t. Then, all requests manufactured in time interval $[t_1, t-1]$ have due dates earlier than t.*

Similar to the previous algorithm we need the following notation. For $i \in [1..n]$ and time periods $t_1 \leq t_2$, such that $d_i \in [t_1, t_2]$, let $O(i, t_1, t_2) = \{O_j \mid j \geq i \wedge d_j \in [t_1, t_2]\}$. That is, $O(i, t_1, t_2)$ is the set of all requests at least as important as O_i whose due date is within the time interval $[t_1, t_2]$.

The dynamic programming table Π of size $O(n^2 \cdot T^2)$ is defined as follows. The entry $\Pi(i, t_1, t_2, k)$, where $i \in [1..n]$, and $d_i \in [t_1, t_2]$, contains the minimum production cost of the requests in $O(i, t_1, t_2)$ in time interval $[t_1, t_2]$ with the additional constraint that the production capacity at time t_1 is $k \leq c(t_1)$ and if $k < c(t_1)$ then the (residual) cost of production at t_1 is zero.

Let $d_{\min}$ and $d_{\max}$ be the the minimum and maximum due dates. Then the optimal solution is given by $\Pi(1, d_{\min}, d_{\max}, c(d_{\min}))$.

The base cases are similar to the ones described in Section 2. We show how to compute an entry $\Pi(i, t_1, t_2, k)$, for $t_1 < t_2$ and $k > 0$ in polynomial time given all entries $\Pi(i', t_1', t_2', k')$ with $i' > i$. We enumerate over all possible times to produce O_i. Suppose that O_i is produced at time t. This partitions the problem into two subproblems: one with all requests in $O(i, t_1, t_2)$ whose due date is before t, and all the rest of the requests (excluding O_i) that by Observation 2 must be manufactured no earlier than time t.

$$\Pi(i, t_1, t_2, k) = \min_{\substack{t \in [t_1, \min\{d_i, t_2\}] \\ \wedge c(t) > 0}} \left\{ \Pi(i', t_1, t-1, k) + \Pi(i'', t, t_2, k'') + w'(t) + H_i(t) \right\} \tag{2}$$

where:

- $i' = \min\{j : j > i \wedge t_1 \leq d_j < t\}$
- $i'' = \min\{j : j > i \wedge t \leq d_j \leq t_2\}$
- $k'' = \begin{cases} c(t) - 1 & t > t_1 \\ k - 1 & t = t_1 \end{cases}$

- $w'(t) = \begin{cases} w(t) & t > t_1 \\ w(t) & t = t_1 \wedge k = c(t) \\ 0 & \text{otherwise} \end{cases}$

Finally, we note that if the requests are not unit size but polynomially bounded, we can treat each request as a collection of unit size requests and apply the same algorithms to get an optimal solution. (In this case we assume that production can be split over several time periods.) As we have already mentioned, for requests with arbitrary size (that are not polynomially bounded) the problem is NP-hard, even if there is only a single request [8].

4 Fractional rectangle stabbing

In this section we present LP relaxations of d-dimensional rectangle stabbing with soft and hard capacities. The constraints in these LPs are not restricted to packing or covering constraints. Hence, solving the LPs in polynomial time requires general LP algorithms (such as the Ellipsoid algorithm or interior point methods). We then show that the LP relaxations can be interpreted as network flow problems if some of the variables are fixed.

4.1 LP formulation

Following [5], we consider the linear programming relaxation for HARD-d-RS. To simplify notation we write $u \in S$ instead of $u \in \mathcal{U}(S)$.

$$\min \sum_{S \in \mathcal{S}} x(S)$$

$$\text{s.t.} \sum_{S \mid u \in S} y(S,u) \geq 1 \qquad \forall u \in \mathcal{U} \tag{3}$$

$$\sum_{u \in S} y(S,u) \leq c(S)x(S) \qquad \forall S \in \mathcal{S} \tag{4}$$

$$y(S,u) \leq x(S) \qquad \forall S,u \tag{5}$$

$$x(S) \leq 1 \qquad \forall S \in \mathcal{S} \tag{6}$$

$$x(S), y(S,u) \geq 0 \qquad \forall S,u \tag{7}$$

We denote this LP by LP-HARD. The variable $x(S)$ indicates the "portion" of S that belongs to the cover. The variable $y(S,u)$ indicates the portion of u that is covered by S. Constraints of type (3) are simply covering constraints. Capacity constraints are formulated using constraints of types (4) and (5). Constraints of type (6) and type (7) are fractional relaxations of $x(S), y(S,u) \in \{0,1\}$. Note that there is a variable $y(S,u)$ only if $u \in S$. However, to simplify notation, we consider all pairs (S,u), regardless of whether $u \in S$. In case $u \notin S$, we simply assign $y(S,u) = 0$.

An LP-relaxation of SOFT-d-RS is obtained by omitting constraints of type (6). We denote the LP-relaxation without constraints of type (6) by LP-SOFT.

The integrality gap of both LP-HARD and LP-SOFT in the d-dimensional case is $\Omega(d)$, even without capacities. Consider a d-RS instance that contains $n = \binom{d}{\lceil d/2 \rceil}$ (d-dimensional) rectangles and d hyperplanes of dimension $d-1$. All these hyperplanes cut through the origin, and for $i = 1, \ldots, d$, the ith hyperplane is orthogonal to the ith axis. For each subset of $\lceil \frac{d}{2} \rceil$ hyperplanes there is a single rectangle that intersects exactly the hyperplanes in this subset. For example, the

rectangle that intersects exactly the $\lceil \frac{d}{2} \rceil$ hyperplanes each of which is orthogonal to one of the first $\lceil \frac{d}{2} \rceil$ axes, can be defined by the points

$$R = \{(x_1, \ldots, x_d) \mid -1 \leq x_i \leq 1, \text{ for } 1 \leq i \leq \lceil d/2 \rceil \text{ and } 1 \leq x_i \leq 2, \text{ for } \lceil d/2 \rceil < i \leq d\}.$$

Any integral solution must set the value of at least 1 to at least $\lceil \frac{d}{2} \rceil$ hyperplanes. Otherwise, at least $\lceil \frac{d}{2} \rceil$ hyperplanes are unselected resulting in at least one uncovered rectangle. On the other hand, it is not difficult to see that a fractional solution that sets the value $\frac{1}{\lceil d/2 \rceil}$ to each rectangle is feasible. The integrality gap follows.

Note also that the integrality gap of both LP-HARD and LP-SOFT is at least $2 - o(1)$ even in the one-dimensional case. Consider an instance that contains $k+1$ rectangles and two lines of capacity k that intersect all the rectangles. A fractional optimal solution is $x^*(S) = \frac{k+1}{2k}$ for each line S and $y^*(S,u) = \frac{1}{2}$ for every line S and rectangle u. This means that the value of the fractional minimum is $1 + \frac{1}{k}$, while the integral optimum is 2.

The following definitions apply to both LP-HARD and LP-SOFT. We refer to a pair (x, y) as a *partial cover* if it satisfies all the constraints, except (perhaps) constraints of type (3). A rectangle is *covered* if its type (3) constraint is satisfied. If $\sum_{S \mid u \in S} y(S,u) \geq \alpha$, we refer to u as *α-covered*. If $\sum_{S \mid u \in S} y(S,u) > 0$ we say that u is *positively covered*.

We denote an optimal solution by (x^*, y^*). The sum $\sum_{S \in \mathcal{S}} x^*(S)$ is denoted by OPT*. Without loss of generality we assume that the covering constraints are tight, i.e., that $\sum_{S \mid u \in S} y^*(S,u) = 1$ for every $u \in \mathcal{U}$.

4.2 A network flow formulation

This section is written in HARD-d-RS terms, but similar arguments can be made in the case of SOFT-d-RS. Once the values of the x variables are fixed, it is very useful to view the LP relaxation as a network flow problem [4, 5]. Namely, we are given a (fractional) set of lines x and wish to find the best possible assignment y.

The network N_x is the standard construction used for bipartite graphs (see Fig. 2 for an example). On one side we have all the lines and on the other side we have all the rectangles. There is an arc (S,u) if $u \in S$. The capacity of an arc (S,u) equals $x(S)$. There is a source s that feeds all the lines. The capacity of each arc (s,S) emanating from the source equals $x(S) \cdot c(S)$. There is a sink t that is fed by all the rectangles. The capacity of every arc (u,t) entering the sink equals 1.

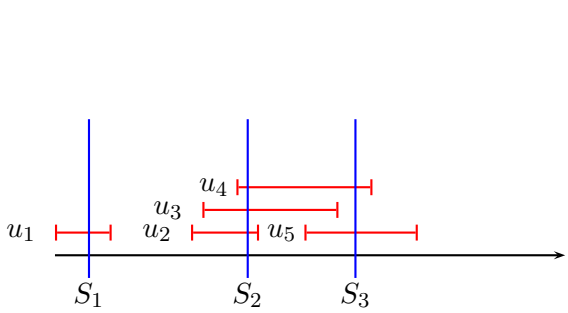
In the one dimensional case, the bipartite graph described above is a *convex* bipartite graph (see [15, p.196]).

Observation 3. *There is a one-to-one correspondence between vectors y such that (x,y) is a partial cover and flows f in N_x. The correspondence $y \leftrightarrow f_y$ satisfies $f_y(u,t) = \sum_{S \mid u \in S} y(S,u)$, for every rectangle $u \in \mathcal{U}$, and $f_y(s,S) = \sum_{u \in S} y(S,u)$, for every line $S \in \mathcal{S}$.*

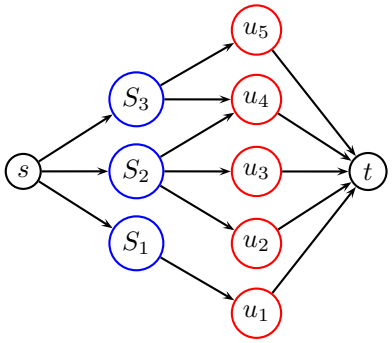
Proof. Given y simply define f_y as follows.

$$f_y(e) \triangleq \begin{cases} \sum_{u \in S} y(S,u) & \text{if } e = (s,S), \\ y(S,u) & \text{if } e = (S,u), \\ \sum_{S \mid u \in S} y(S,u) & \text{if } e = (u,t). \end{cases}$$

The mapping from flows to vectors is defined similarly. ■



(a) A HARD-1-RS instance (capacities are omitted).



(b) The corresponding network.

Figure 2: A HARD-1-RS instance and the corresponding network.

We often refer to $f_y(s, S)$ as the *flow supplied by* S and to $f_y(u, t)$ as the *flow delivered to* u. To simplify notation, we denote $f_y(s, S)$ by $f_y(S)$ and $f_y(u, t)$ by $f_y(u)$. We say that y is *maximum with respect to* x if f_y is a maximum flow in N_x.

Next, we show that we can identify infeasible instances of HARD-d-RS.

Observation 4. *Feasibility of a* HARD-d-RS *instance can be verified by computing a maximum integral flow in a network* N_x*, where* $x(S) = 1$*, for every* $S \in \mathcal{S}$*.*

The following observation is based on the integrality of a max-flow in a network with integral capacities. It implies that it suffices to compute a feasible cover (x, y), where x is integral.

Observation 5 ([5])**.** *Let* (x, y) *be a feasible solution of* LP-HARD*. If* x *is integral, then an integral* y' *such that* (x, y') *is a feasible solution can be computed in polynomial time.*

Definition 6. *Let* (x, y) *and* (x, y') *be partial covers. We say that* y' dominates y *if (i)* $f_{y'}(u) \geq f_y(u)$*, for every* $u \in \mathcal{U}$*, and (ii)* $f_{y'}(S) \geq f_y(S)$*, for every* $S \in \mathcal{S}$*. We write* $y' \succeq y$ *to denote that* y' *dominates* y*.*

Observation 7. *Let* (x, y) *denote a partial cover. Then one can find in polynomial time a vector* y' *that satisfies: (i)* y' *is maximum with respect to* x*, and (ii)* y' *dominates* y*.*

Proof. We use an augmenting path algorithm to compute a maximum flow f' in N_x starting with f_y. The flow f' induces the desired vector $y' \succeq y$ since saturating an augmenting path from s to t never decreases the flow in edges exiting s, or in edges entering t. ■

Let **aug-flow** be an efficient algorithm that given a partial cover (x, y), finds a vector $y' \succeq y$ that is maximum with respect to x. Note that **aug-flow** may change the assignment of lines to rectangles. In terms of the network flow, the flow of certain edges may decrease, but the sum of flows that enters (exits, respectively) every rectangle (line, respectively) does not decrease.

5 d-dimensional rectangle stabbing

In this section we present a $3d$-approximation algorithm for SOFT-d-RS. We also present a bi-criteria approximation algorithm for HARD-d-RS that computes $4d$-approximate cover that uses at most two copies of each line. The algorithms are based on solving LP-SOFT or LP-HARD, and then rounding the solution. For the sake of simplicity, the algorithms are presented for the two dimensional case ($d = 2$).

5.1 Finding a partial cover

Let (x^*, y^*) be an optimal solution of LP-SOFT. We define

$$H \triangleq \{S \,|\, x^*(S) > \varepsilon\} \qquad \text{and} \qquad L \triangleq \{S \,|\, x^*(S) \leq \varepsilon\} \ .$$

Let $L = L^h \cup L^v$ denote a partition of L into horizontal and vertical lines. Loosely speaking, we partition the horizontal lines in L^h into "contiguous blocks" by accumulating lines in L^h from "top" to "bottom" until the sum of fractional values $x^*(S)$ in the block is exactly ε. A block may start (or end) in the "middle" of a line S. In this case, S belongs to two consecutive blocks, and we treat S as though it has two copies, one belongs to the left block and the other to the right block. Also, $x^*(S)$ is split between the copies.

Formally, let $L^h = \{S_1, \ldots, S_m\}$. Assign the segment[1] $I_1 = [0, x^*(S_1))$ to S_1, and assign the segment $I_j = [\sum_{i<j} x^*(S_i), \sum_{i \leq j} x^*(S_i))$ to S_j. Partition L^h into blocks $L^h_1, L^h_2, \ldots, L^h_{b(h)}$ and the (possibly empty) leftover block $\tilde{L}^h$. The block L^h_i contains the lines S_j whose segment I_j has a nonempty intersection with the segment $[(i-1)\cdot\varepsilon, i\cdot\varepsilon)$, and the block $\tilde{L}^h$ contains the lines that intersect $[b(h)\cdot\varepsilon, (b(h)+1)\cdot\varepsilon)$. Note that the intersection of every two consecutive blocks contains at most one line. Such a line is split into two copies.

By the construction, $\sum_{S \in L^h_j} x^*(S) = \varepsilon$ for every $i \leq b(h)$ and $\sum_{S \in \tilde{L}^h} x^*(S) < \varepsilon$. The same type of partitioning is applied to the vertical lines in L^v to obtain the blocks $L^v_1, \ldots, L^v_{b(v)}$ and the leftover block $\tilde{L}^v$.

Observation 8. *The number of blocks (not including the leftover block) in each dimension satisfies* $b(h) \leq \frac{1}{\varepsilon} \cdot \sum_{S \in L^h} x^*(S)$ *and* $b(v) \leq \frac{1}{\varepsilon} \cdot \sum_{S \in L^v} x^*(S)$.

Let $S^*_{h,j}$ and $S^*_{v,j}$ denote lines of maximum capacity in L^h_j and L^v_j, respectively. Let

$$L^* \triangleq \left\{S^*_{h,j} \,|\, 1 \leq j \leq b(h)\right\} \cup \left\{S^*_{v,j} \,|\, 1 \leq j \leq b(v)\right\}.$$

We note that the two copies of the same line may be chosen by two consecutive blocks.

Definition 9. *We define the partial cover (x, y) as follows. The support of the cover is $H \cup L^*$. For every $S \in H$ and $u \in \mathcal{U}(S)$, we keep $x(S) = x^*(S)$ and $y(S, u) = y^*(S, u)$. For every $S \in L^*$ and $u \in \mathcal{U}(S)$, let $B(S)$ denote the block that contains S. Then,*

$$x(S) = \sum_{S' \in B(S)} x^*(S') = \varepsilon \qquad \textit{and} \qquad \textit{for } u \in S, \quad y(S,u) = \sum_{S' \in B(S)} y^*(S', u).$$

Note that if $u \in S^*_{h,j}$, then $S^*_{h,j}$ covers u to the same extent that u is covered by lines in L^h_j according to y^*. Hence, rectangles that are intersected by $S^*_{h,j}$ are "locally satisfied". Also notice that $\sum_S x(S) \leq \sum_S x^*(S)$. We now prove that (x, y) is a indeed partial cover.

[1] We use the term *segment* instead of the term *interval* to avoid confusion.

Claim 10. *(x, y) is a partial cover.*

Proof. We first show that constraints of type (5) are satisfied, namely, that $y(S, u) \leq x(S)$, for every u, S. Clearly, this is true for $S \notin L^*$. Consider a line $S^* \in L^*$. Let B denote the block of lines in L that contains S^*. For every rectangle u intersected by S^*, the following holds:

$$y(S^*, u) = \sum_{S' \in B} y^*(S', u) \leq \sum_{S' \in B} x^*(S') = x(S^*).$$

We now show that constraints of type (4) are satisfied, namely, that $\sum_{u \in S} y(S, u) \leq c(S)x(S)$, for every $S \in \mathcal{S}$. This trivially holds for $S \notin H \cup L^*$ since both $x(S) = 0$, and $y(S, u) = 0$. Constraint (4) holds for $S \in H$, since $x(S) = x^*(S)$, and $y(S, u) = y^*(S, u)$. It remains to consider a line $S^* \in L^*$. Let B denote the block of lines in L that contains S^*.

$$\begin{aligned}
\sum_{u \in S^*} y(S^*, u) &= \sum_{u \in S^*} \sum_{S \in B} y^*(S, u) \\
&\leq \sum_{S \in B} \sum_{u \in S} y^*(S, u) \\
&\leq \sum_{S \in B} c(S) x^*(S) \\
&\leq \max_{S \in B} c(S) \cdot \sum_{S \in B} x^*(S) \\
&= c(S^*) \cdot x(S^*) \ .
\end{aligned}$$

The first inequality follows from the fact that some rectangles may lose part of their flow, the second inequality is due to LP Constraints of type (4), and the third inequality follows from Definition 9. ■

Claim 11. *The coverage of every rectangle u is greater than $(1 - 2d\varepsilon)$ in the partial cover (x, y).*

Proof. Consider a rectangle u. We show that, in each dimension, the coverage of u decreases by less than 2ε due to the transition from y^* to y. By definition, coverage by lines in H is preserved. In addition, if a rectangle u intersects all the lines in a block L^h_j, then the coverage of u by lines in L^h_j is now covered by $S^*_{h,j}$. Namely, $\sum_{S \in L^h_j} y^*(S, u) = y(S^*_{h,j}, u)$. It follows that u may lose coverage only in the "leftmost" and "rightmost" blocks that u intersects. In each such block, the coverage of u is bounded by ε, (a one dimensional example is given in Fig. 3). Since u is covered in (x^*, y^*), it follows that $\sum_S y(S, u) > 1 - d \cdot 2\varepsilon$, and the claim follows. ■

5.2 Rectangle stabbing with soft capacities

In this section we show how to obtain $3d$-approximate solutions for SOFT-d-RS.

First, by replicating lines, we modify the fractional solution (x^*, y^*) so that $x^*(S) \leq 1/3$ for every S. If there exists a line S such that $x^*(S) > 1/3$ we replace it with $k = \lceil 3x^*(S) \rceil$ copies of S denoted by $S_1, \ldots, S_k$, where $x^*(S_i) = x^*(S)/k$ and $y^*(S_i, u) = y^*(S, u)/k$ for every $u \in S$. Obviously, feasibility and cost is preserved by the splitting of S into k copies.

Let $\varepsilon = \frac{1}{3d}$. By Claim 11, each rectangle is 1/3-covered by (x, y). A cover is obtained by scaling by a factor of 3, namely $x'(S) = \lceil 3x(S) \rceil = 1$ for every $S \in \mathcal{S}$ such that $x(S) > 0$ and $y'(S, u) = 3y(S, u)$ for every $u \in S$. Note that if $x(S) > 0$ then $x(S) \geq \varepsilon = \frac{1}{3d}$, and thus $x'(S) \leq 3d \cdot x(S)$. Clearly, every rectangle is covered by (x', y'). Moreover, by Observation 5 an integral y'' such that (x', y'') is a cover can by computed in polynomial time. (x', y'') is a $3d$-approximation, since $x'(S) \leq 3d \cdot x(S)$ for every S, and $\sum_S x(S) \leq \sum_S x^*(S)$.

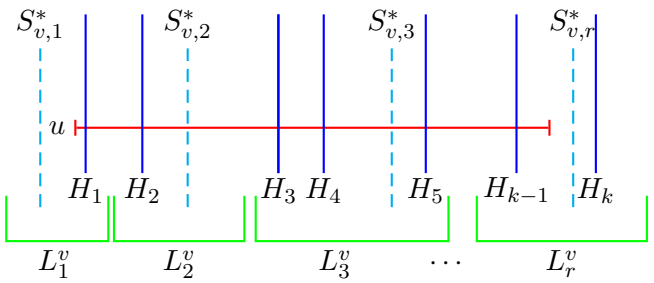


Figure 3: An interval u covered by both lines in H and by lines in L. $S^*_{v,1}$ and $S^*_{v,r}$ do not intersect u, and therefore, the flow supplied to u by f_{y^*} in the blocks L_1 and L_r is lost.

5.3 Rectangle stabbing with hard capacities

We present a bi-criteria approximation algorithm for HARD-d-RS that computes a $4d$-approximate cover that uses at most two copies of each line.

We first compute an optimal solution for LP-HARD. Afterwards, we set $\varepsilon = \frac{1}{4d}$ and compute H and L^* using the algorithm that was presented in Section 5.1. By Claim 11 we get that each rectangle is 1/2-covered by (x, y). A cover is obtained by scaling as follows. Let $x'(S) = \lceil 2x(S) \rceil$ for every $S \in \mathcal{S}$, and $y'(S, u) = 2y(S, u)$ for every $S \in \mathcal{S}$ and $u \in S$. Note that $x(S) \leq 1$, for every line $S \in H$, hence $x'(S) = \lceil 2 \cdot x(S) \rceil \leq 2$. Also, note that every line $S \in L^*$ may belong to at most two blocks, and that for each such occurrence $x'(S) = 1$. Therefore, S may appear at most twice in the solution. Also, note that we rely on Observation 5 to compute an integral y'' such that (x', y'') is a cover.

The approximation ratio of $4d$ is proved as follows. First, note that $x(S) > 0$ only if $S \in H \cup L^*$. Now, if $S \in H$ and $x(S) > 1/2$, then $x'(S) = 2 \leq 4x(S)$. Otherwise, $x(S) \leq 1/2$ and therefore $x'(S) = 1 \leq 4d \cdot x(S)$. It follows that $x'(S) \leq 4d \cdot x(S)$ for every S, which means that (x', y'') is a $4d$-approximate cover.

6 Hardness results

In this section we present two hardness results. Both results rely on the fact that Set Cover cannot be approximated within a factor of $c \log n$ for some $c > 0$, unless P=NP [7, 17]. The first reduction shows that the HARD-d-RS problem is Set Cover hard, for $d \geq 2$, if weights are given to the lines. The second reduction shows that it is NP-hard to approximate d-RS within $c \log d$, for some constant c when the dimension d is part of the input.

6.1 Weighted rectangle stabbing with hard capacities

We show that for $d \geq 2$ the weighted capacitated rectangle stabbing is as hard to approximate as Set Cover. Chuzhoy and Naor [5] presented a reduction of Set Cover to weighted capacitated Vertex Cover in bipartite graphs. Our reduction mimics their reduction by applying it to the adjacency matrix of bipartite graphs (as in the Egerváry-König Theorem).

Consider the Set Cover instance with the collection $\mathcal{C} = \{C_1, \ldots, C_m\}$ of subsets of $\{1, \ldots, n\}$. We construct the two-dimensional instance $(\mathcal{S}, \mathcal{U})$ of weighted capacitated rectangle stabbing as follows. The rows, indexed $1, \ldots, n$ correspond the elements, and the columns, indexed $1, \ldots, m$ correspond to the subsets. The set of rectangles $\mathcal{U}$ contains a unit (one by one) square u_{ij} for every pair (i, C_j) such that $i \in C_j$. The coordinates of the center of u_{ij} are (i, j). The vertical lines in $\mathcal{S}$ are $(x = j)$, for $1 \leq j \leq m$. Each vertical line is assigned unit weight and capacity n. The horizontal lines in $\mathcal{S}$ are the lines $(y = i)$, for $1 \leq i \leq n$. The weight of every horizontal line is zero, and the capacity of the line $(y = i)$ is $|\{C_j \mid i \in C_j\}| - 1$.

Given a solution $\mathcal{C}' \subseteq \mathcal{C}$ to the Set Cover instance, the corresponding solution to the rectangle stabbing instance consists of the vertical lines $(x = j)$, where $C_j \in \mathcal{C}'$ and all the horizontal lines. Given a solution $\mathcal{S}' \subseteq \mathcal{S}$ to the rectangle stabbing instance, it is easy to see that $\{C_j \mid (x = j) \in \mathcal{S}'\}$ is a Set Cover. This completes the approximation preserving reduction.

6.2 Rectangle stabbing with dimension d

We present an approximation preserving reduction from Set Cover to Rectangle Stabbing. The dimension of the reduced instance equals the number of sets in the Set Cover instance. Therefore, it is NP-hard to approximate d-RS within $c \log d$, for some constant c.

Consider the Set Cover instance with the collection $\mathcal{C} = \{C_1, \ldots, C_m\}$ of subsets of $\{1, \ldots, n\}$. We construct the following instance of m-dimensional rectangle stabbing. For every $1 \leq j \leq n$, let $\chi_j \in \mathbb{R}^m$ be a vector that indicates which subsets in $\mathcal{C}$ contain j. Formally, $\chi_j = (\chi_{j1}, \ldots, \chi_{jm})$, where $\chi_{ji} = 1$ if $j \in S_i$, and zero otherwise. For every $1 \leq j \leq n$, let u_j denote the rectangle whose opposite corners are χ_j and $(-1, \ldots, -1)$. Each set C_i is represented by the hyperplane $x_i = 1/2$ (i.e., the ith coordinate equals $1/2$). Since a rectangle u_j intersects a hyperplane $(x_i = 1/2)$ if and only if $j \in C_i$, the reduction follows.

7 Open problems and concluding remarks

We list a few open problems. It is unknown whether there exist $O(d)$-approximation algorithms for HARD-d-RS or for weighted SOFT-d-RS. (We showed that weighted HARD-d-RS is set-cover hard.)

Gaur et al. [10] presented a d-approximation algorithm for weighted d-RS that uses linear programming to reduce the problem into d one-dimensional instances. The analysis of their algorithm relies on the integrality of the LP relaxation in the one dimensional case. Our exact algorithm for weighted SOFT-1-RS does not imply a bound on the integrality gap. On the other hand, our 3-approximation algorithm for unweighted SOFT-1-RS implies that the integrality gap of unweighted SOFT-1-RS is at most 3. Hence another $3d$-approximation algorithm follows by combining a reduction similar to the Gaur et al. [10] and our 3-approximation algorithm for SOFT-1-RS.

Acknowledgment

We thank Alexander Ageev for pointing out an error in an earlier version of the paper.

References

[1] S. Anily and M. Tzur. Shipping multiple-items by capacitated vehicles – an optimal dynamic programming approach. *Transportation Science*, 39:233–248, 2005.

[2] S. Anily, M. Tzur, and L. A. Wolsey. Multi-item lot-sizing with joint set-up cost. Submitted for publication, 2005.

[3] P. Baptiste. Scheduling unit tasks to minimize the number of idle periods: a polynomial time algorithm for offline dynamic power management. In *17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 364–367, 2006.

[4] J. Bar-Ilan, G. Kortsarz, and D. Peleg. Generalized submodular cover problems and applications. *Theoretical Computer Science*, 250:179–200, 2001.

[5] J. Chuzhoy and J. Naor. Covering problems with hard capacities. *SIAM Journal on Computing*, 36(2):498–515, 2006.

[6] G. Even, D. Rawitz, and S. Shahar. Approximation algorithms for capacitated rectangle stabbing. In *6th International Conference on Algorithms and Complexity*, volume 3998 of *LNCS*, pages 18–29, 2006.

[7] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[8] M. Florian, J. K. Lenstra, and A. H. G. Rinooy Kan. Deterministic production planning: Algorithms and complexity. *Management Science*, 26:669–679, 1980.

[9] R. Gandhi, E. Halperin, S. Khuller, G. Kortsarz, and A. Srinivasan. An improved approximation algorithm for vertex cover with hard capacities. *Journal of Computer and System Sciences*, 72(1):16–33, 2006.

[10] D. R. Gaur, T. Ibaraki, and R. Krishnamurti. Constant ratio approximation algorithms for the rectangle stabbing problem and the rectilinear partitioning problem. *Journal of Algorithms*, 43:138–152, 2002.

[11] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[12] S. Guha, R. Hassin, S. Khuller, and E. Or. Capacitated vertex covering. *Journal of Algorithms*, 48(1):257–270, 2003.

[13] R. Hassin and N. Megiddo. Approximation algorithms for hitting objects with straight lines. *Discrete Applied Mathematics*, 30(1):29–42, 1991.

[14] S. Kovaleva and F. C. R. Spieksma. Approximation of rectangle stabbing and interval stabbing problems. In *12th European Symposium on Algorithms*, volume 3221 of *LNCS*, pages 426–435, 2004.

[15] E. Lawler. *Combinatorial Optimization: networks and matroids*. Courier Dover Publications, 2001.

[16] R. Levi, A. Lodi, and M. Sviridenko. Approximation algorithms for the multi-item capacitated lot-sizing problem via flow-cover inequalities. In *12th Conference on Integer Programming and Combinatorial Optimization*, volume 4513 of *LNCS*, pages 454–468, 2007.

[17] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *29th ACM Symposium on the Theory of Computing*, pages 475–484, 1997.

[18] S. Shahar. *Approximation Algorithms for Problems with Geometrical Structure*. PhD thesis, School of Electrical Engineering, Tel-Aviv University, 2006.

[19] L. A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2:385–393, 1982.