Journal of Algorithms **43,** 138–152 (2002)
doi:10.1006/jagm.2002.1221, available online at http://www.idealibrary.com on IDEAL®

Constant Ratio Approximation Algorithms for the Rectangle Stabbing Problem and the Rectilinear Partitioning Problem

Daya Ram Gaur

School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada V5A 1S6
E-mail: gaur@cs.sfu.ca

Toshihide Ibaraki

Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto, 606-8501, Japan
E-mail: ibaraki@i.kyoto-u.ac.jp

and

Ramesh Krishnamurti

School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada V5A 1S6
E-mail: ramesh@cs.sfu.ca

Received June 29, 2001

We provide constant ratio approximation algorithms for two NP-hard problems, the rectangle stabbing problem and the rectilinear partitioning problem. In the rectangle stabbing problem, we are given a set of rectangles in two-dimensional space, with the objective of stabbing all rectangles with the minimum number of lines parallel to the x and y axes. We provide a 2-approximation algorithm, while the best known approximation ratio for this problem is $O(\log n)$. This algorithm is then extended to a 4-approximation algorithm for the rectilinear partitioning problem, which, given an $m_x \times m_y$ array of nonnegative integers and positive integers v, h, asks to find a set of v vertical and h horizontal lines such that the maximum load of a subrectangle (i.e., the sum of the numbers in it) is minimized. The best known approximation ratio for this problem is 27. Our approximation ratio 4 is close to the best possible, as it is known to be NP-hard to approximate within any factor less than 2. The results are then extended to the d-dimensional space for $d \geq 2$, where a

0196-6774/02 $35.00

d-approximation algorithm for the stabbing problem and a d^d-approximation algorithm for the partitioning problem are developed.

Key Words: rectangle stabbing; rectilinear partitioning; combinatorial optimization; approximation algorithms.

1. INTRODUCTION

We consider in this paper two NP-hard problems, the rectangle stabbing problem and the rectilinear partitioning problem. An important application of the rectilinear partitioning problem arises when a two-dimensional array of processors connected as a mesh is used in areas such as numeric computation and image processing. The computation is done over an array of numbers, where each number represents computation load. The underlying algorithm may involve finite element methods or two-dimensional finite difference equations for numeric computation or iterative computation such as convolution for image processing (see, e.g., [7]). The problem is to map the numbers (or loads) to the processors, so as to minimize the total load (sum of the numbers) assigned to one processor.

When the processors in a parallel system are connected as a two-dimensional mesh, a useful method to map the numbers to the processors is to first perform a rectilinear partitioning of the two-dimensional array of numbers, in which a set of v vertical and h horizontal lines are drawn to subdivide the array of numbers. Each subrectangle (subarray of numbers enclosed by successive vertical and horizontal lines) is now assigned to a processor in the mesh. This type of mapping is particularly useful when communication between adjacent processors on the mesh (adjacent in a row or column) is fast compared to communication between non-adjacent processors. In a mapping using rectilinear partitioning, adjacent data in the array are always mapped to adjacent processors, thereby reducing the communication overhead. It is noted that the rectilinear partitioning method can be applied for dimensions other than two. As an example, given a one-dimensional array of data points and a one-dimensional array of processors, the problem is to allocate consecutive data points to processors to optimize some criterion [5]. The d-dimensional partitioning problem using $(d-1)$-dimensional hyperplanes, for $d \geq 2$, will be discussed in Section 4.2 of this paper.

In the above applications, either the same instruction stream applies to all processors (in the SIMD model of computation), or computation proceeds iteratively, where in each iteration every processor finishes its computation before the next iteration is started (accomplished by a synchronization step in the MIMD model of computation). In both these models of computation, it is important to minimize the maximum load on a processor, where the

load on a processor is the sum of the computation loads assigned to the processor. The load for a mapping is therefore defined as the largest load among the loads on all the processors. This largest load determines the execution time (for an instruction in SIMD and for an iteration in MIMD) and is thus the bottleneck in the total execution time of the program. The objective is therefore to seek a mapping that minimizes the largest load.

In this paper, we study the rectilinear two-dimensional data partitioning problem. Berger and Bokhari [3] look at the partitioning problem where the two-dimensional array is partitioned into rectangles such that the load is balanced (each rectangle has the same load). However, the partitioning they investigate is not by straight lines, but by those having right-angle bends. Then, they investigate the cost of mapping such a partition onto the set of processors connected as a mesh, as a tree, and as a hypercube. Bokhari [4] discusses the one-dimensional version and provides an $O(n^3 p)$ algorithm to obtain the optimal load for n data points and p processors. A better algorithm with running time $O(pn \log n)$ is proposed in [17], which is later improved to $O(pn)$ using a dynamic programming formulation [5]. For an excellent treatise on the rectilinear partitioning problem, the reader is referred to Nicol's paper [16]. It provides an $O(n + (p \log n)^2)$ algorithm for the one-dimensional partitioning problem and proves that the three-dimensional rectilinear partitioning problem is NP-hard. It states that the complexity of the two-dimensional rectilinear partitioning problem is open.

More recent work on this problem establishes the NP-hardness of the two-dimensional problem [9]. The $\Theta(\sqrt{p})$ approximation algorithm that was provided for this problem [1] was later improved, when a simple approximation algorithm with a performance ratio of 27 was provided [13]. Aspvall *et al.* [1] point out that the NP-hardness proof in [9] implies that it is NP-hard to approximate the problem within any factor less than 2. Both the approximation algorithms assume that the number of horizontal lines equals the number of vertical lines. In this paper, we provide an approximation algorithm with a performance ratio of 4, for an $m_x \times m_y$ array of data points, which is to be partitioned by h ($\leq m_x$) horizontal lines and v ($\leq m_y$) vertical lines.

Our algorithm for the rectilinear partitioning problem is based on a reduction to the problem of stabbing a set of rectangles by the minimum number of lines parallel to the axes. The latter problem has been studied before by Hassin and Megiddo, for which the best known approximation algorithm has a performance ratio of $O(\log n)$ for n rectangles [10]. The algorithm we provide in this paper has an approximation ratio of 2.

These results in this paper are then extended to the d-dimensional space for $d \geq 2$, where stabbing and partitioning are done by $(d-1)$-dimensional hyperplanes. For the stabbing problem, we have a d-approximation algorithm,

and for the rectilinear partitioning problem, we have a d^d-approximation algorithm.

In Section 2, we discuss the rectangle stabbing problem (denoted RS) and provide a 2-approximation algorithm. In Section 3, we proceed to the rectilinear partitioning problem (denoted RP), and use the 2-approximation algorithm for RS to obtain a 4-approximation algorithm for RP. In Section 4, we consider various extensions of the results, including those for d-dimensions.

2. AN APPROXIMATION ALGORITHM FOR RECTANGLE STABBING

We formally define problem RS, and then provide a 2-approximation algorithm.

Problem RS. We are given a set R of n rectangles, which are aligned to the x and y axes in the two-dimensional space, whose corners are all located at integer points and whose areas are more than 1 (i.e., unit squares are excluded). Using at least h horizontal lines and v vertical lines, respectively, the problem is to stab all the rectangles by the minimum number of lines located at integer coordinates.

A horizontal (resp., vertical) line is said to *stab* a rectangle r if it goes through its interior. Since all corners of r are integer points, a horizontal line $y = c$ stabs r if $a + 1 \leq c \leq b - 1$ holds for the coordinate $y = a$ (resp., $y = b$) of the lower edge (resp., upper edge) of r, where a, b, c are all restricted to be integers. A similar argument also holds for vertical lines. Unit squares are not permitted in set R because it is not possible to stab them in the above sense.

The NP-hardness of problem RS was established by Hassin and Megiddo [10]. This problem is a special case of the set covering problem and the hitting set problem [6, 11, 12]. The set covering problem and the hitting set problem admit no approximation algorithm with a constant performance ratio [2, 14], where the performance ratio is the worst-case ratio of the solution value returned by the algorithm to the optimum value. The best known approximation ratio for RS is $O(\log n)$ [10].

We begin by describing the integer and linear programming formulations for RS. Let H be the set of all horizontal lines whose coordinates are one larger than those of the lower edges of the given rectangles or whose coordinates are one smaller than those of the upper edges:

$$H = \{y = a + 1, y = b - 1 \mid y = a \text{ is the lower edge and } y = b \text{ is the upper edge of a rectangle } r \in R \text{ such that } b - a \geq 2\}.$$

Correspondingly, let V be the set of all vertical lines whose coordinates are one larger than those of the left edges of the given rectangles or whose coordinates are one smaller than those of the right edges:

$$V = \{x = a + 1, x = b - 1 \mid x = a \text{ is the left edge and } x = b \text{ is the right edge of a rectangle } r \in R \text{ such that } b - a \geq 2\}.$$

To solve problem RS, we can assume without loss of generality that lines are chosen from set $H \cup V$, since any horizontal or vertical line can be shifted to a line in $H \cup V$ without changing the stabbing condition of all the rectangles.

The cardinalities of H and V are at most $2n$, respectively. Let x_i be the decision variable corresponding to the ith horizontal line in the set H, and let y_j be the decision variable corresponding to the jth vertical line in V. For each rectangle $r_k, k \in \{1, 2, \ldots, n\}$, let H_k be the set of horizontal lines in H that stab rectangle r_k, and let V_k be the set of vertical lines in V that stab rectangle r_k. Permitting slight abuse of notations, we also write $i \in H$ and $j \in V$ to denote the ith horizontal line in H and the jth vertical line in V. Similarly, we write $i \in H_k$ and $j \in V_k$ to denote the ith horizontal line in H_k and the jth vertical line in V_k. Finally we let $[n]$ represent the set $\{1, 2, \ldots, n\}$.

Now consider the following integer program P and its linear programming relaxation $\overline{P}$.

P	$\overline{P}$
$\min \sum_{i \in H} x_i + \sum_{j \in V} y_j$	$\min \sum_{i \in H} x_i + \sum_{j \in V} y_j$
$\sum_{i \in H_k} x_i + \sum_{j \in V_k} y_j \geq 1, \quad k \in [n]$	$\sum_{i \in H_k} x_i + \sum_{j \in V_k} y_j \geq 1, \quad k \in [n]$
$\sum_{i \in H} x_i \geq h$	$\sum_{i \in H} x_i \geq h$
$\sum_{j \in V} y_j \geq v$	$\sum_{j \in V} y_j \geq v$
$x_i, y_j \in \{0, 1\}, \quad i \in H, \quad j \in V$	$x_i, y_j \geq 0, \quad i \in H, \quad j \in V$

Note that any feasible solution to the integer program P above defines a set of lines that stab all the rectangles, and an optimal solution to P gives an optimal solution to RS. In particular, if a solution exists with h horizontal

lines and v vertical lines that stab all rectangles, then the optimal solution to the above program P will indeed return such a solution.

Now let $(\bar{x}_i, i \in H; \bar{y}_j, j \in V)$ be an optimal fractional solution to problem $\overline{P}$. Such a solution satisfies

$$\text{either } \sum_{i \in H_k} \bar{x}_i \geq 1/2 \qquad \text{or} \qquad \sum_{j \in V_k} \bar{y}_j \geq 1/2$$

for every rectangle $k \in [n]$. Let R_H (resp., R_V) be the set of all k for which $\sum_{i \in H_k} \bar{x}_i \geq 1/2$ (resp., $\sum_{j \in V_k} \bar{y}_j \geq 1/2$) holds. Based on these, we introduce the following integer programs P_H and P_V.

P_H	P_V
$\min \sum_{i \in H} x_i$	$\min \sum_{j \in V} y_j$
$\sum_{i \in H_k} x_i \geq 1, \quad k \in R_H$	$\sum_{j \in V_k} y_j \geq 1, \quad k \in R_V$
$\sum_{i \in H} x_i \geq h$	$\sum_{j \in V} y_j \geq v$
$x_i \in \{0, 1\}, \qquad i \in H$	$y_j \in \{0, 1\}, \qquad j \in V$

It should be noted that every constraint in integer programs P_H and P_V contains a subset of variables in the corresponding constraint of P. Let a feasible solution to P_H be x_H, and let a feasible solution to P_V be y_V. It is then easy to see that the composite solution (x_H, y_V) is feasible in P.

We denote the linear programming relaxations of P_H and P_V (i.e., $x_i \in \{0, 1\}$ is relaxed to $x_i \geq 0$, and $y_j \in \{0, 1\}$ is relaxed to $y_j \geq 0$) by $\overline{P}_H$ and $\overline{P}_V$, respectively. Then we denote the optimal values for the programs $P, \overline{P}, P_H, P_V, \overline{P}_H$, and $\overline{P}_V$ by $P^*, \overline{P}^*, P_H^*, P_V^*, \overline{P}_H^*$, and $\overline{P}_V^*$, respectively. The following lemmas provide relationships among these values.

LEMMA 1. *$P_H^* = \overline{P}_H^*$ and $P_V^* = \overline{P}_V^*$.*

Proof. We first consider P_H and order all the horizontal lines $i \in H$ from the lowest to the highest. Then all the horizontal lines in H_k of each k are consecutive in this ordering. Thus the rows of the coefficient matrix of P_H, which correspond to the first set of constraints for all $k \in R_H$, have the interval property (i.e., the elements 1's are located consecutively in each row), and such a matrix is totally unimodular (e.g., pp. 540–544, Corollary 2.10, and Proposition 2.1 in [15]). Thus any fractional optimal solution to $\overline{P}_H$ is integral and solves P_H, proving that $P_H^* = \overline{P}_H^*$. A similar argument also applies to P_V and $\overline{P}_V$. ∎

LEMMA 2. $P_H^* + P_V^* \leq 2P^*$.

Proof. Let $(\bar{x}, \bar{y})$ be an optimal fractional solution to problem $\overline{P}$. We observe that the solutions $(2\bar{x})$ and $(2\bar{y})$, where every variable value is multiplied by 2, provide a feasible solution to $\overline{P}_H$ and $\overline{P}_V$, respectively. This is true because, for every constraint of $\overline{P}_H$ for $k \in R_H$, $\sum_{i \in H_k} \bar{x}_i \geq 1/2$ holds and hence $\sum_{i \in H_k} 2\bar{x}_i \geq 1$. Furthermore, $\sum_{i \in H} 2\bar{x}_i \geq h$ is obvious for $\overline{P}_H$ since $\sum_{i \in H} \bar{x}_i \geq h$ holds by definition. Thus $(2\bar{x})$ is feasible to $\overline{P}_H$. A similar argument can also be applied to $\overline{P}_V$. Consequently it follows that $\overline{P}_H^* + \overline{P}_V^* \leq 2\overline{P}^*$. This, together with Lemma 1 and $\overline{P}^* \leq P^*$, proves the lemma statement. ∎

The following theorem gives us the desired performance ratio.

THEOREM 1. *There exists a polynomial time* 2*-approximation algorithm for problem RS.*

Proof. From Lemmas 1 and 2, it is evident that the following algorithm APPROX_RS has approximation ratio 2 for problem RS. It requires us to solve three linear programs, $\overline{P}$, $\overline{P}_H$, and $\overline{P}_V$, which can obviously be done in polynomial time. ∎

Algorithm APPROX_RS

Input: A set R of n rectangles aligned to the x and y axes, and nonnegative integers h and v.

Output: h' $(\geq h)$ horizontal lines and v' $(\geq v)$ vertical lines that together stab all rectangles in R.

1. *Solve linear program* $\overline{P}$*, and obtain the fractional optimal solution* $(\bar{x}, \bar{y}) = (\bar{x}_i, i \in H; \bar{y}_j, j \in V)$.

2. *Construct the two integer programs* P_H *and* P_V *from* $(\bar{x}, \bar{y})$*, and obtain their optimal solutions* x_H^* *and* y_V^* *(by solving linear programs* $\overline{P}_H$ *and* $\overline{P}_V$ *(see Lemma* 1*)), respectively.*

3. *Output the ith horizontal lines such that* $(x_H^*)_i = 1$ *and the jth vertical lines such that* $(y_V^*)_j = 1$.

The time complexity of APPROX_RS is dominated by the time to solve three linear programs in Steps 1 and 2. As our linear programs have 0–1 coefficient matrices, they can be solved in strongly polynomial time (i.e., polynomial in the numbers of variables and constraints) [18]. The time complexity for the algorithm is $O(n^5)$ for n rectangles.

COROLLARY 1. *If an instance of RS has a solution with h horizontal lines and v vertical lines, then APPROX_RS outputs a solution with h' horizontal lines and v' vertical lines such that $h \le h' \le 2h$ and $v \le v' \le 2v$.*

Proof. In this case, any optimal solution $(\bar{x}, \bar{y})$ to $\overline{P}$ has objective values $\sum_{i \in H} \bar{x}_i = h$ and $\sum_{j \in V} \bar{y}_j = v$ in $\overline{P}_H$ and $\overline{P}_V$, respectively. Hence the argument in the proof of Lemma 2 shows that $h' = P_H^* \le 2h$ and $v' = P_V^* \le 2v$. The remaining properties $h \le h'$ and $v \le v'$ are obvious. ∎

3. THE RECTILINEAR PARTITIONING PROBLEM

We define the rectilinear partitioning problem (denoted RP) in this section and provide a 4-approximation algorithm. We are given an $m_x \times m_y$ array A of nonnegative integers $a_{ij}, 1 \le i \le m_x, 1 \le j \le m_y$. We are also given the numbers h of horizontal lines and v of vertical lines. A horizontal line can only be placed between two successive rows of the array, and a vertical line can only be placed between two successive columns. For convenience, we regard the lower boundary (resp., upper boundary) of A as the 0th (resp., $(h+1)$-st) horizontal line. Similarly, the left boundary (resp., right boundary) of A is regarded as the 0th (resp., $(v+1)$-st) vertical line. Given the pth and $(p+1)$-st horizontal lines, and the qth and $(q+1)$-st vertical lines, we define the rectangle r_{pq} as comprising all the numbers a_{ij} enclosed by the four lines. We define the load of r_{pq} by

$$L_{pq} = \sum_{a_{ij} \in r_{pq}} a_{ij}.$$

Problem RP. Given an $m_x \times m_y$ array $A = \{a_{ij}, 1 \le i \le m_x, 1 \le j \le m_y\}$ of nonnegative integers, and nonnegative integers h and v, place h horizontal lines and v vertical lines such that the largest load $\max\{L_{pq} \mid 0 \le p \le h, 0 \le q \le v\}$ is minimized.

Before providing an approximation algorithm for RP, we introduce the following decision problem version of RP (denoted DRP).

Problem DRP. Given an $m_x \times m_y$ array A of nonnegative integers and nonnegative integers h, v and L, decide whether there are h horizontal lines and v vertical lines such that $\max_{p,q} L_{pq} \le L$ holds.

If we had an algorithm for solving DRP exactly, we could solve RP optimally by finding the minimum L such that the corresponding DRP outputs 'YES.' To find such L, the value of L may be initially set to

$$a_{\text{sum}} = \sum_{i,j} a_{ij},$$

and a binary search is conducted over the interval $[0, a_{\text{sum}}]$. As DRP is NP-complete, however, we solve it only approximately by the following algorithm, which uses APPROX_RS in the previous section as a subalgorithm.

Algorithm APPROX_DRP

Input: An $m_x \times m_y$ array A of nonnegative integers and nonnegative integers h, v, and L.

Output: 'YES' together with h' horizontal lines and v' vertical lines such that $h \le h' \le 2h$, $v \le v' \le 2v$, and $\max_{p,q} L_{pq} \le L$ hold; else 'NO.'

1. *Generate all rectangles r in A such that $\sum_{a_{ij} \in r} a_{ij} > L$, and denote the resulting set as R. This defines an instance of problem* RS.

2. *Call* APPROX_RS *to obtain a set of h' ($\ge h$) horizontal lines and v' ($\ge v$) vertical lines that stab all the rectangles in R.*

3. *If $h' \le 2h$ and $v' \le 2v$, then return* 'YES,' *together with the h' horizontal lines and v' vertical lines obtained in Step* 2*; else return* 'NO.'

In Step 1, we transform an instance of problem DRP into an instance of problem RS by constructing the set of all rectangles R in array A such that every rectangle $r \in R$ satisfies $\sum_{a_{ij} \in r} a_{ij} > L$. As each rectangle r can be specified by two elements of A located at its upper left corner and its lower right corner and there are $m_x m_y$ elements in A, R contains at most $m_x^2 m_y^2$ rectangles. Therefore, by considering each of these rectangles systematically, it is not difficult to see that Step 1 can be done in $O(m_x^2 m_y^2)$ time.

The heart of the above algorithm is Steps 2 and 3. Corollary 1 tells that, if the instance of problem RS defined in Step 1 has a solution with h horizontal lines and v vertical lines, the APPROX_RS outputs h' ($\le 2h$) horizontal lines and v' ($\le 2v$) vertical lines, and hence APPROX_DRP returns 'YES.' In this case, the largest load of a subrectangle generated by the solution of APPROX_DRP is at most L, because otherwise there exists a rectangle r with $\sum_{a_{ij} \in r} a_{ij} > L$, which is not stabbed by any of the lines in the solution, a contradiction.

These are summarized as follows.

LEMMA 3. (a) *If an instance of* DRP *has a solution, then* APPROX_DRP *returns* 'YES.'

(b) *If* APPROX_DRP *returns* 'YES,' *then the associated output satisfies* $h \le h' \le 2h$, $v \le v' \le 2v$, *and* $\max_{p,q} L_{pq} \le L$.

Algorithm APPROX_DRP calls algorithm APPROX_RS with at most $m_x^2 m_y^2$ rectangles as input. Thus, algorithm APPROX_DRP also runs in strongly polynomial time. In the following algorithm APPROX_RP, we find the smallest L for which APPROX_DRP returns 'YES,' by repeatedly calling APPROX_DRP. By using the fact that if APPROX_DRP returns 'NO' for the current L, then it returns 'NO' for all $L' \le L$; the smallest L for which APPROX_DRP returns 'YES' can be obtained by performing a binary search over the interval $[0, a_{\text{sum}}]$. This takes $\lceil \log_2 a_{\text{sum}} \rceil$ iterations. Given the solution for this smallest $L = L_{\min}$, we then reduce the numbers of horizontal lines h' and vertical lines v' to h and v, respectively. This can be done by removing every second horizontal and vertical line, until the number of horizontal lines becomes equal to h and the number of vertical lines becomes equal to v. After this modification, it is easy to observe that the load of each of the resulting subrectangles is at most $4L_{\min}$.

Algorithm APPROX_RP

Input: An $m_x \times m_y$ array A of nonnegative integers, and nonnegative integers h and v.

Output: h horizontal lines and v vertical lines.

1. *Apply a binary search over* $[0, a_{\text{sum}}]$ *to find the minimum* L (*denoted* $L_{\min}$) *such that* APPROX_DRP *returns* 'YES.'

2. *Consider the output of* APPROX_DRP *for* $L_{\min}$ (*i.e.,* h' *horizontal lines and* v' *vertical lines*). *Noting that* $h' \le 2h$ *and* $v' \le 2v$ *hold, remove alternate horizontal and vertical lines, respectively, until the resulting set contains* h *horizontal lines and* v *vertical lines. Then output the remaining horizontal and vertical lines.*

THEOREM 2. *Algorithm* APPROX_RP *outputs a feasible solution of problem RP, and its approximation ratio is* 4.

Proof. Let the optimal load for problem RP be L_{opt}. Then, by Lemma 3, APPROX_DRP for $L = L_{\text{opt}}$ returns 'YES' using at most $2h$ horizontal lines and $2v$ vertical lines. Therefore, the $L_{\min}$ found by APPROX_RP satisfies $L_{\min} \le L_{\text{opt}}$. This implies that, by removing every alternate line among the horizontal and vertical lines, the load computed by algorithm APPROX_RP is at most $4L_{\min}$ ($\le 4L_{\text{opt}}$). Thus the approximation ratio as

stated in the theorem follows. It is also obvious that the output contains h horizontal lines and v vertical lines; hence it is feasible to RP. ∎

Algorithm APPROX_RP calls algorithm APPROX_DRP at most $O(\log_2 a_{\text{sum}})$ times and therefore also runs in strongly polynomial time.

4. EXTENSIONS

We consider two types of extensions of problems RS and RP in the following sections.

4.1. *Total Number of Horizontal and Vertical Lines*

Our results readily extend to a version of RS in which there are no lower bounds on the number of horizontal and vertical lines. Consider that we are given a set of n rectangles aligned to the axes in two-dimensional space, and the objective is to minimize the total number of lines parallel to the axes, which stab all the given rectangles. For this, we delete the constraints $\sum_{i \in H} x_i \geq h$ and $\sum_{j \in V} y_j \geq v$ from the formulations $P, \overline{P}, P_H$, and P_V. It is easily verified that the approximation ratio stays the same.

Similarly, we can extend our results to the total number version of problem RP. That is, given an input array A and a nonnegative integer t, it asks to find a total of t horizontal and vertical lines that minimize the maximum load $\max_{p,q} L_{pq}$. Based on the approximate algorithm for the above version of RS, it is straightforward to modify algorithm APPROX_RP so that the same approximation ratio 4 holds for this problem.

4.2. *Higher Dimensions*

All the discussion so far can be extended to the d-dimensional space with $d \geq 2$. Let d coordinates be denoted $x_l, l \in [d]$, and assume that a set R of n d-dimensional rectangles $r_k, k \in [n]$, are given. Each d-rectangle r_k is the intersection of $2d$ halfspaces $x_l \geq a_{kl1}$ and $x_l \leq a_{kl2}, l \in [d]$, where a_{klj} are all assumed to be nonnegative integers. A hyperplane $x_l = b$ then stabs a rectangle r_k if $a_{kl1} < b < a_{kl2}$ holds for the given l. The d-dimensional counterpart of problem RS, denoted d-RS, is then defined as follows.

Problem d-RS. Given a set R of n d-rectangles, as described above, and d nonnegative integers $h_l, l \in [d]$, we are asked to stab all the rectangles with the minimum number of hyperplanes under the constraint that at least h_l hyperplanes of type $x_l = b$ are used for each $l \in [d]$.

Now define the sets of hyperplanes,

$$H_l = \{x_l = a_{kl1} + 1, \quad x_l = a_{kl2} - 1 \mid k \in [n]\}, \qquad l \in [d]$$

and

$$H_{kl} = \{i \in H_l \mid \text{the } i\text{th hyperplane in } H_l \text{ stabs } r_k\}, \qquad k \in [n],\ l \in [d],$$

in which we again permit slight abuse of notations such as $i \in H_l$.

Then introducing 0–1 variables y_{li} to denote if the ith hyperplane in H_l is selected ($y_{li} = 1$) or not ($y_{li} = 0$) in the solution, we formulate problem d-RS as the following integer program:

$$\begin{aligned} d\text{-}P: \min & \sum_{l\in[d]} \sum_{i\in H_l} y_{li} \\ & \sum_{l\in[d]} \sum_{i\in H_{kl}} y_{li} \geq 1, \qquad k \in [n] \\ & \sum_{i\in H_l} y_{li} \geq h_l, \qquad l \in [d] \\ & y_{li} \in \{0,1\}, \qquad i \in H_l, l \in [d]. \end{aligned}$$

The continuous relaxation of d-P, denoted d-$\overline{P}$, is then defined by replacing constraints $y_{li} \in \{0,1\}$ with $y_{li} \geq 0$. Let $(\bar{y}_l, l \in [d])$ denote the optimal fractional solution of d-$\overline{P}$, where $(\bar{y}_l) = (\bar{y}_{li}, i \in H_l)$. For each $k \in [n]$, it is obvious that at least one $l \in [d]$ satisfies

$$\sum_{i\in H_{kl}} \bar{y}_{li} \geq 1/d.$$

For each $l \in [d]$, let R_l be the set of all $k \in [n]$ for which the above inequality holds. Similarly to P_H and P_V in Section 2, we introduce the following d integer programs d-$P_l, l \in [d]$:

$$\begin{aligned} d\text{-}P_l: \min & \sum_{i\in H_l} y_{li} \\ & \sum_{i\in H_{kl}} y_{li} \geq 1, \qquad k \in R_l \\ & \sum_{i\in H_l} y_{li} \geq h_l \\ & y_{li} \in \{0,1\}, \qquad i \in H_l. \end{aligned}$$

We denote the continuous relaxation of d-P_l by d-$\overline{P}_l$, which is defined similarly to d-$\overline{P}$.

Then the argument in Section 2 can be directly generalized to this d-dimensional case by using the solution $(d\bar{y}_l, l \in [d])$ in place of $(2\bar{x}, 2\bar{y})$, and we have the next result.

THEOREM 3. *For a given dimension d ($\geq$2), there exists a polynomial time d-approximation algorithm for problem d-RS.*

Such an approximation algorithm may be described as follows.

Algorithm APPROX_d-RS

Input: A set R of n d-rectangles aligned to d axes $x_l, l \in [d]$ and nonnegative integers $h_l, l \in [d]$.

Output: h'_l $(\geq h_l)$ hyperplanes of type $x_l = b, l \in [d]$, which together stab all rectangles in R.

1. *Solve program* $\overline{P}$, *and obtain the fractional optimal solution* $(\bar{y}_l, l \in [d])$.

2. *Construct the* d *integer programs* d-$P_l, l \in [d]$, *from* $(\bar{y}_l, l \in [d])$, *and obtain their optimal solutions* $(y^*_l), l \in [d]$ (*by solving linear programs* d-$\overline{P}_l, l \in [d]$), *respectively.*

3. *Output the ith hyperplane in* H_l *such that* $y^*_{li} = 1$ *for* $i \in H_l$ *and* $l \in [d]$.

COROLLARY 2. *If an instance of* d-*RS has a solution with* h_l *hyperplanes of type* $x_l = b, l \in [d]$, *then APPROX_*d-*RS outputs a solution with* h'_l *hyperplanes*, $l \in [d]$, *such that* $h_l \leq h'_l \leq dh_l$ *holds for all* $l \in [d]$.

Finally we introduce the extended version of problem RP.

Problem d-*RP.* Given a d-dimensional array $A = \{a_{i_1 i_2 \cdots i_d} \mid 1 \leq i_1 \leq m_1, 1 \leq i_2 \leq m_2, \ldots, 1 \leq i_d \leq m_d\}$ and nonnegative integers $h_l, l \in [d]$, place h_l hyperplanes $x_l = b_{lj}, j \in [h_l]$ such that the largest load $\max\{L_{p_1 p_2 \cdots p_d} \mid 0 \leq p_l \leq h_l, l \in [d]\}$ is minimized, where $L_{p_1 p_2 \cdots p_d}$ is the sum of loads $a_{i_1 i_2 \cdots i_d}$ in the d-rectangle enclosed by p_lth and $(p_l + 1)$-st hyperplanes of type $x_l = b$ for all $l \in [d]$.

The decision problem version d-DRP of d-RP is also defined similarly to DRP in Section 2. For these, algorithms APPROX_DRP and APPROX_RP can be naturally generalized to the d-dimensional space, resulting in algorithms APPROX_d-DRP and APPROX_d-RP, respectively, though we omit their details. In this process of defining algorithms, we only need to replace conditions such as $h' \leq 2h$ and $v' \leq 2v$ by conditions $h'_l \leq dh_l, l \in [d]$. In algorithm APPROX_$d$-RP, the hyperplanes of type $x_l = b, l \in [d]$ are chosen in such a manner that $(d - 1)$ hyperplanes are removed after each retained one.

As a result, we have the following extension of Theorem 2.

THEOREM 4. *Algorithm* APPROX_d-RP *outputs a feasible solution of problem* d-*RP, and its approximation ratio is* d^d.

Algorithm APPROX_d-RP runs in time $O(dn^5)$, where n is the number of d-dimensional rectangles.

5. CONCLUSION

In this paper, we provided improved approximation algorithms for two NP-hard problems. An important result of the paper is a 2-approximation algorithm for the rectangle stabbing problem. This improves on the log n-approximation result, the best known so far. We then used this algorithm to improve the approximation ratio for the rectilinear partitioning problem. We provided a 4-approximation algorithm, which improves on the best-known approximation ratio of 27. Considering that it is NP-hard to approximate the problem to within any factor less than 2 [1, 9], we are within a factor 2 of the lower bound on the approximation ratio for this problem. These results for the stabbing problem and the partitioning problem are then extended to the general d-dimensional space.

ACKNOWLEDGMENTS

A preliminary version of this paper was presented in ESA2000 [8]. During the development of our algorithms, the authors have benefited from discussions with various people. In particular, Kaz Makino of Osaka University gave us many useful comments, Magnus Halldórsson of University of Iceland pointed out the argument about the lower bound on the approximation ratio for problem RP in [1, 9], and Alexander A. Ageev of Institute of Mathematics, Novosibirsk, independently found the extendibility to the d-dimensional space and discovered a number of interesting problems for which the argument in this paper is applicable. The second author was partially supported by a Grant-in-Aid from the Ministry of Education, Science, Sports and Culture of Japan. The third author would like to acknowledge support from the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

1. B. Aspvall, M. M. Halldórsson, and F. Manne, Approximations for the general block distribution of a matrix, *in* "Proceedings of the Fifth Scandinavian Workshop on Algorithm Theory," Lecture Notes in Computer Science, Vol. 1432, pp. 47–58. Springer-Verlag, Berlin/New York, 1998.
2. M. Bellare, S. Goldwasser, C. Lund, and A. Russell, Efficient probabilistically checkable proofs and applications to approximation, *in* "Proceedings of the 25th ACM Symposium on Theory of Computation," pp. 294–304, 1993.
3. M. J. Berger and S. H. Bokhari, A partitioning strategy for nonuniform problems on multiprocessors, *IEEE Trans. Comput.* **C-36**, No. 5 (1987), 570–580.
4. S. H. Bokhari, Partitioning problems in parallel, pipelined, and distributed computing, *IEEE Trans. Comput.* **C-37**, No. 1 (1988), 48–57.

5. H. A. Choi and B. Narahari, Algorithms for mapping and partitioning chain structured parallel computations, *in* "Proceedings of the International Conference on Parallel Processing," Vol. 1, pp. 625–628, 1991.
6. V. Chvátal, A greedy heuristic for the set covering problem, *Math. Oper. Res.* **4** (1979), 233–235.
7. G. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker, "Solving Problems on Concurrent Processors: General Techniques and Regular Problems," Prentice Hall, Englewood Cliffs, NJ, 1988.
8. D. R. Gaur, T. Ibaraki, and R. Krishnamurti, Constant ratio approximation algorithms for the rectangle stabbing problem and the rectilinear partitioning problem, *in* "Proceedings of the Eighth Annual European Symposium of Algorithms," Lecture Notes in Computer Science, Vol. 1879, pp. 211–219, Springer-Verlag, Berlin/New York, 2000.
9. M. Grigni and F. Manne, On the complexity of the generalized block distribution, *in* "Proceedings of the Third International Workshop on Parallel Algorithms for Irregularly Structured Problems," Lecture Notes in Computer Science, Vol. 1117, pp. 319–326, Springer-Verlag, Berlin/New York, 1996.
10. R. Hassin and N. Megiddo, Approximation algorithms for hitting objects with straight lines. *Discrete Appl. Math.* **30** (1991), 29–42.
11. D. S. Hochbaum, Approximation algorithms for the set covering and vertex cover problem, *SIAM J. Comput.* **11** (1982), 555–556.
12. D. S. Johnson, Approximation algorithms for combinatorial problems, *J. Comput. Syst. Sci.* **9** (1974), 256–278.
13. S. Khanna, S. Muthukrishnan, and S. Skiena, Efficient array partitioning, *in* "Proceedings of the 24th International Colloquium on Automata, Languages and Programming," Lecture Notes in Computer Science, Vol. 1256, pp. 616–626, Springer-Verlag, Berlin/New York, 1997.
14. C. Lund and M. Yannakakis, On the hardness of approximating minimization problems, *J. ACM* **41** (1994), 960–981.
15. G. Nemhauser and L. Wolsey, "Integer and Combinatorial Optimization," Wiley–Interscience, New York, 1988.
16. D. M. Nicol, Rectilinear partitioning of irregular data parallel computations, *J. Parallel Distributed Comput.* **23** (1994), 119–134.
17. D. M. Nicol and D. R. O'Hallaron, Improved algorithms for mapping parallel and pipelined computations, *IEEE Trans. Comput.* **40**, No. 3 (1991), 295–306.
18. E. Tardos, A strongly polynomial algorithm to solve combinatorial linear programs, *Oper. Res.* **34** (1986), 250–256.