On the Rectangle Escape Problem*

A. Ahmadinejad† S. Assadi‡ E. Emamjomeh-Zadeh§
S. Yazdanbod¶ H. Zarrabi-Zadeh‖

Abstract

Motivated by the bus escape routing problem in printed circuit boards, we study the following *rectangle escape* problem: given a set S of n axis-aligned rectangles inside an axis-aligned rectangular region R, extend each rectangle in S toward one of the four borders of R so that the maximum density over the region R is minimized. The density of each point $p \in R$ is defined as the number of extended rectangles containing p. We show that the problem is hard to approximate to within a factor better than $3/2$ in general. When the optimal density is sufficiently large, we provide a randomized algorithm that achieves an approximation factor of $1+\varepsilon$ with high probability improving over the current best 4-approximation algorithm available for the problem. When the optimal density is one, we develop an exact algorithm that finds an optimal solution efficiently. We also provide approximation algorithms and inapproximability results for a restricted version of the problem where rectangles are allowed to escape toward only a subset of directions.

Keywords Rectangle escape; Approximation algorithms; Randomized rounding; NP-completeness; Inapproximability

1 Introduction

In this paper, we revisit the *rectangle escape* problem [5], motivated by a closely related escape routing problem in printed circuit boards. The problem is formally defined as follows:

*A preliminary version of this work was presented at CCCG 2013 [2]. The research was conducted when the first four authors were in the Department of Computer Engineering at Sharif University of Technology.

†Management Science and Engineering Department, Stanford University, Stanford, CA 94305, United States. Email: ahmadi@stanford.edu.

‡Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, United States. Email: sassadi@cis.upenn.edu.

§Department of Computer Science, University of Southern California, Los Angeles, CA 90089, United States. Email: emamjome@usc.edu.

¶School of Computer Science, Georgia Institute of Technology, Atlanta, GA 30332-0765, United States. Email: yazdanbod@gatech.edu.

‖Department of Computer Engineering, Sharif University of Technology, Tehran 14588-89694, Iran. Email: zarrabi@sharif.edu.

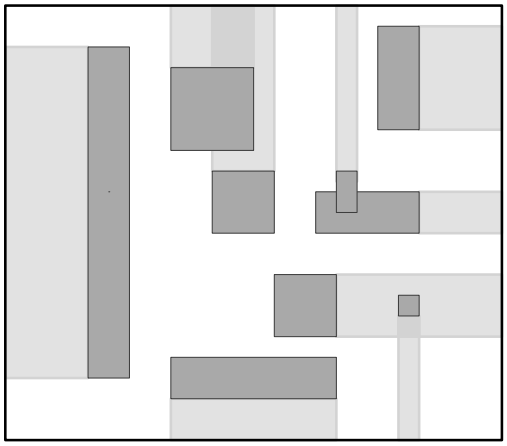


Figure 1: An instance of the rectangle escape problem. The input rectangles are shown in dark gray, and their projections are shown in light gray.

Problem 1 (Rectangle Escape Problem (REP))**.** *Given an axis-parallel rectangular region R, and a set S of n axis-parallel rectangles inside R, extend each rectangle in S toward one of the four borders of R, so that the maximum density over R is minimized, where the density of a point $p \in R$ is defined as the number of extended rectangles containing p.*

To *extend* a rectangle in S, we simply project it onto one of the four borders of R. The bounding box of the rectangle and its projection is called an "extended rectangle". Note that each extended rectangle contains the area of its original rectangle. An example of the rectangle escape problem is illustrated in Figure 1. In this example, the maximum density over the region, which is equal to the maximum number of extended rectangles that overlap at any point is 2. Not that, by our definition of extended rectangles, the density of the points inside the original rectangles is at least one in any solution.

The study of the rectangle escape problem is motivated by the escape routing problem in printed circuit boards (PCBs). The objective in the *escape routing* problem is to route nets from their pins to the boundary of the enclosing component. The problem has been extensively studied in the literature (see, e.g., [3, 4, 5, 6, 7, 9, 10, 11, 12]). In industrial applications, nets are usually grouped into buses, and the nets from each bus are preferred to be routed together. In this model, the routing of a bus is obtained by projecting the bounding box of the bus onto one of the four sides of the bounding component. An example is illustrated in Figure 2. The main objective is to find a bus routing in which the maximum number of overlaps at any point is minimized. This is equivalent to the rectangle escape problem, as defined in Problem 1.

The rectangle escape problem is known to be NP-hard [5]. The *decision version* of the problem, which we call k-REP, is defined as follows:

Problem 2 (k-REP)**.** *Given an instance of the rectangle escape problem and an integer $k \geq 1$, determine whether any routing is possible with a density of at most k.*

It is known that the k-REP problem is NP-complete, even for $k = 3$ [5]. The best current approximation algorithm for the optimization version of the problem is due to Ma and Wong [5]

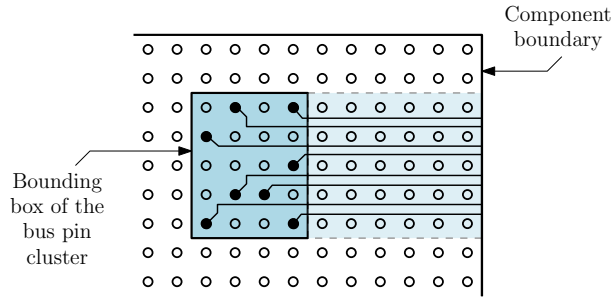


Figure 2: An example of the bus escape routing problem. The routing is obtained by projecting the bounding box of the bus pin cluster onto the component boundary.

that achieves an approximation factor of 4, using a deterministic linear programming (LP) rounding technique.

For a special case when the optimal density is 1 (i.e., when all chips can be routed with no conflict), the problem can be solved exactly using polynomial-time algorithms available for the related *maximum disjoint subset* problem [1, 3].

Our results. In this paper, we obtain some results on the rectangle escape problem, a summary of which is listed below.

- We show that the k-REP problem is NP-complete for any $k \geq 2$. Given that the problem is polynomially solvable for $k = 1$, this fully settles the complexity of the problem for all values of k. An important implication of this result is that the rectangle escape problem is hard to approximate to within any factor better than 3/2, unless P = NP.

- Despite the fact that the problem is hard to approximate to within a constant factor when the optimal density is low, we present a randomized algorithm that achieves an approximation factor of $1+\varepsilon$ with high probability, when the optimal density is at least $c_\varepsilon \log n$, for some constant c_ε. This improves, for instances with high density, upon the current best algorithm of Ma and Wong [5] that guarantees an approximation factor of 4 for all instances. Our algorithm is based on a randomized rounding technique applied to a linear programming formulation of the problem.

- For the 1-REP problem, we present a new algorithm that solves the problem exactly in $O(n^4)$ time, improving upon the previous solution based on the $O(n^6)$-time algorithm of Kong *et al.* [3] for the maximum disjoint subset problem[1]. Our algorithm can indeed solve the following more general optimization version of the 1-REP problem: given an instance of the rectangle escape problem, find a maximum-size subset of rectangles in S that can be routed disjointly.

[1] After the appearance of the preliminary version of this paper, an improved $O(n^4)$-time algorithm is provided in [1] for the maximum disjoint subset problem. The algorithm provided in this paper is still simpler than that of [1].

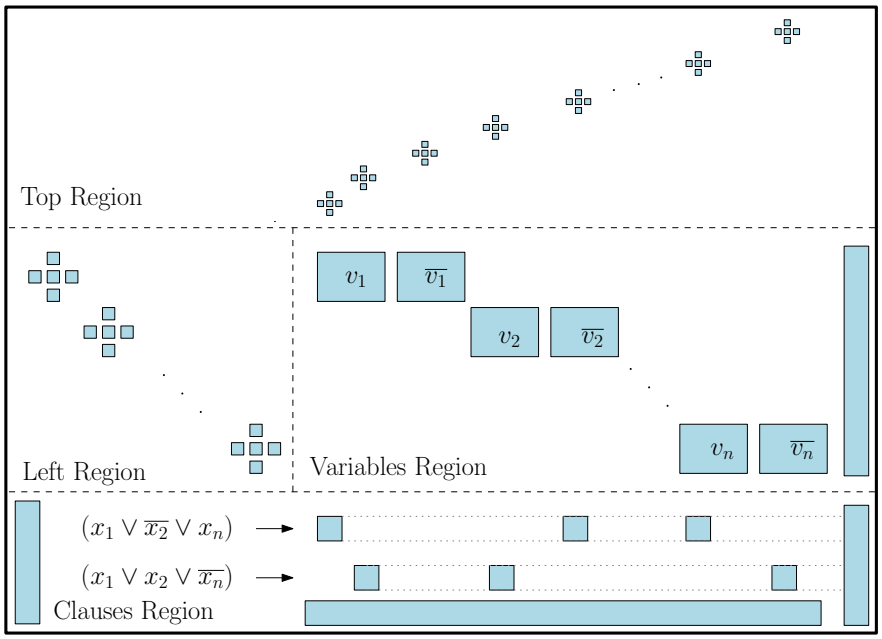


Figure 3: A reduction from 3-SAT to 2-REP.

- We study a restricted version of the REP problem, in which all rectangles are allowed to escape only toward a fixed subset of directions. In particular, we consider the bidirectional REP problem, in which rectangles can only escape toward two adjacent borders of R. We show how this version can be used to approximate the original REP problem, and present some inapproximability results for this restricted version.

2 Hardness Result

We first show that the k-REP problem is NP-complete, for any $k \geq 2$. As a corollary, we show that the rectangle escape problem is hard to approximate to within any factor better than 3/2, unless P = NP. Our hardness result holds even in a more restricted setting where the input rectangles are all disjoint.

Theorem 1. *The k-REP problem is NP-complete for any $k \geq 2$, even if all input rectangles are disjoint.*

Proof. We prove by reduction from 3-SAT. The reduction is similar to that of [5], but requires some non-trivial modifications to handle the special case of $k = 2$, and also, to work in a more restricted setting where all rectangles are disjoint. Given an instance of 3-SAT, we create an instance of 2-REP as follows. Fix a rectangular region R. We partition R into four (virtual) sub-regions, labeled with top, left, variables, and clauses, as shown in Figure 3. Then, we start building a set of rectangles S inside R as follows. We first add one long rectangle to the right side of the variables region, and three long rectangles to the left, right, and bottom sides of the clauses region, as shown in Figure 3. The following rectangles are then added to S.

- For each variable x_i, we add a pair of "variable rectangles" v_i and $\bar{v_i}$ along each other to the variables region in such a way that no two rectangles from different variables can be stabbed by a single horizontal or vertical line.

- For each clause C_j, we add three "literal rectangles" in a horizontal row in the clauses region. Each literal rectangle is placed beneath a variable rectangle corresponding to the literal appeared in the clause. Note that no two literal rectangles intersect, and no two of them can be stabbed by a vertical line.

- For each variable, we add a "block gadget" to the left region, directly to the left of the corresponding variable row. Each gadget is composed of five smaller rectangles in a cross-shape arrangement, as shown in Figure 3. Likewise, for each literal in each clause, we add a block gadget to the top region directly above the corresponding literal rectangle. If a literal appears in no clause, we add a block gadget above the corresponding variable rectangle in the top region. The block gadgets are placed in a way that no two rectangles from different gadgets can be stabbed by a single horizontal or vertical line.

Now, we claim that the answer to the constructed instance of 2-REP is yes if and only if the corresponding 3-SAT instance is satisfiable. First, suppose that the answer to the 2-REP is yes, i.e., there is a proper routing of rectangles with a density of at most 2. We show that there is a satisfying assignment for the 3-SAT instance, in which a literal is set to true (resp., false), if the corresponding variable rectangle is routed rightward (resp., downward). To show this, first observe that for each variable v_i, the two variable rectangles v_i and $\bar{v_i}$ cannot be routed simultaneously to the right, because otherwise, they will cause a density of 3 on the rectangle located to the right side of the variables region. Moreover, for each gadget in the top and the left region, the density over at least one of the gadget rectangles is more than one, and hence, in a proper routing of rectangles, no variable rectangle can be routed neither to the top, nor to the left side.

For each clause, observe that none of its three literal rectangles can escape upward because of the block gadgets in the top region, and no two of them can escape simultaneously to neither left nor right, because of the rectangles put on the left and the right sides of the clauses region. Therefore, at least one literal rectangle from each clause must be routed downward. Furthermore, notice that if a variable rectangle escapes downward, none of the literal rectangles below it can be routed downward, because of the rectangle put at the bottom side of the clauses region.

Now, given a proper routing of the 2-REP instance, we set variable v_i in the 3-SAT instance to 1 if rectangle v_i escape to the right, otherwise, we set it to 0. Note that rectangles for v_i and $\bar{v_i}$ cannot simultaneously escape to the right, so this assignment is feasible. Moreover, for each clause, at least one of its literal rectangles, say x_i, must escape downward, meaning that its corresponding variable x_i is set to 1 for sure, and thus the clause is satisfied. Therefore, the 3-SAT instance is satisfiable. The opposite side can be proved using the same exact mapping, and taking into account the fact that there is a proper routing for the top and the left gadget rectangles, in which they do not interfere with the rectangles in the variables and the clauses regions. This completes the NP-completeness proof for $k = 2$.

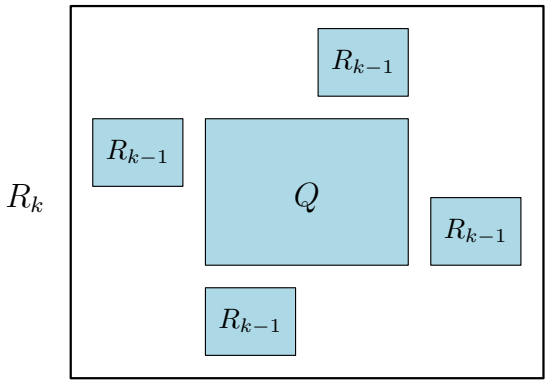


Figure 4: Constructing an instance of k-REP from four instances of $(k-1)$-REP.

To show NP-completeness for $k > 2$, we use a recursive construction. Let R_{k-1} be an instance of $(k-1)$-REP. We construct an instance R_k of k-REP by putting a large rectangle Q in the middle, and four instances of R_{k-1} around Q, as shown in Figure 4. The four instances are placed in a way that no horizontal or vertical line can simultaneously stab any two of them. Now, suppose that R_k has a proper routing of density k. In this routing, Q escapes to one of the four directions, and hence, one of the R_{k-1} instances must have a proper routing of density $k-1$. Therefore, the corresponding 3-SAT instance is satisfiable by induction.

For the opposite side, we show by induction that given a satisfiable instance of k-SAT, the answer to the corresponding R_k is yes. To this end, we strengthen the induction hypothesis as follows: given a satisfiable instance of k-SAT, there is a routing for the corresponding R_k such that (i) the density over R_k is at most k, (ii) the density over the top border of R_k is at most $k-1$, and (iii) the density over the other borders of R_k is at most 1. It can be easily verified that this is true for $k = 2$. Namely, given a satisfiable instance of 3-SAT, we route the rectangles in the corresponding R_2 as follows: we route all variable and literal rectangles to either left, right, or down, as explained before. For each block gadget, we route the squares in the middle and the bottom of the gadget leftward, and route the other squares upward. Finally, we route the long rectangles on the left and the right sides of clauses region downward, the long rectangle on the bottom side of clauses region rightward, and the long rectangle on the right side of variables region upward. This routing guarantees that the maximum density over all borders of the region is at most 1. Now, assume that for $k \geq 2$, the rectangles in the four R_{k-1} instances of R_k can be routed under the above conditions. Namely, for each of the R_{k-1} instances, the density over the top border is at most $k-2$, and the density over the other borders is at most 1. It is easy to see that the maximum density over the initial location of Q is at most k. Therefore, if Q escapes upward, the resulting routing for R_k satisfies the above three conditions. □

As a corollary of Theorem 1, we obtain the following inapproximablity result.

Theorem 2. *For any $\alpha < 3/2$, there is no α-approximation algorithm for the rectangle escape problem, even if all input rectangles are disjoint, unless $P = NP$.*

Proof. Suppose by way of contradiction that there is an algorithm with an approximation factor of $\alpha < 3/2$. If we run this algorithm on an instance of the rectangle escape problem

with an optimal density of 2, the algorithm must return a solution with density less than $3/2 \times 2$, which is at most 2 due to the integrality of the density. Such an algorithm solves the 2-REP problem exactly, which is a contradiction. □

3 An Exact Algorithm for Unit Density

In this section, we present a dynamic programming algorithm that solves the 1-REP problem optimally in $O(n^4)$ time. This is an improvement upon the previous solution due to Kong *et al.* [3] that requires $O(n^6)$ time. Our algorithm is also simpler than the $O(n^4)$-time algorithm provided in [1] for a more general problem of finding a maximum disjoint subset of boundary rectangles. Our algorithm indeed solves the following optimization problem, which we call *maximum disjoint routing*.

Problem 3 (Maximum Disjoint Routing)**.** *Given an instance of the rectangle escape problem with disjoint rectangles, find the maximum number of rectangles that can be routed disjointly, i.e., with unit density.*

It is easy to observe that any algorithm for Problem 3 can also solve 1-REP: we first find the maximum number of rectangles that can be routed disjointly, and check if this number is equal to n. Note that in the above definition, the initial locations of unescaped rectangles are also important: an escaped rectangle cannot collide with any other rectangle, even if that rectangle is not escaped.

Let $R_1, \ldots, R_n$ be the input rectangles, sorted in decreasing order of the y-coordinates of their bottom sides. For a rectangle R_i, the direction $d \in \{$*left, right, up, down*$\}$ is said to be *free* if by escaping toward that direction, R_i does not collide with any other rectangle in its initial place. Note that the freeness of direction d for R_i is independent of the escaping direction of other rectangles. Furthermore, we define the set $\{v_1, \ldots, v_k\}$ $(k \leq 2n)$ as the set of all vertical lines obtained by extending the vertical sides of the rectangles, sorted from left to right.

To solve Problem 3, we first solve two simpler cases in which the escaping directions are only vertical. Given integers $0 \leq i \leq n$ and $1 \leq l, r \leq k$, we define the following two subroutines:

- One-Direction(i, l, r): returns the maximum number of rectangles among $R_1, \ldots, R_i$ that are between v_l and v_r and can be routed upward in unit density.
- Two-Directions(i, l, r): returns the maximum number of rectangles among $R_1, \ldots, R_i$ that are between v_l and v_r and can be routed either upward or downward in unit density.

For each triple (i, l, r), the value of both One-Direction(i, l, r) and Two-Directions(i, l, r) can be calculated by the following simple greedy algorithm. For each rectangle R_j $(1 \leq j \leq i)$ between v_l and v_r, find a free direction upward (and downward, depending on the subproblem). If such direction exists, route R through that direction. Note that routing a rectangle vertically poses no additional restriction on other rectangles in these two subproblems. Next, we define the following additional subproblem.

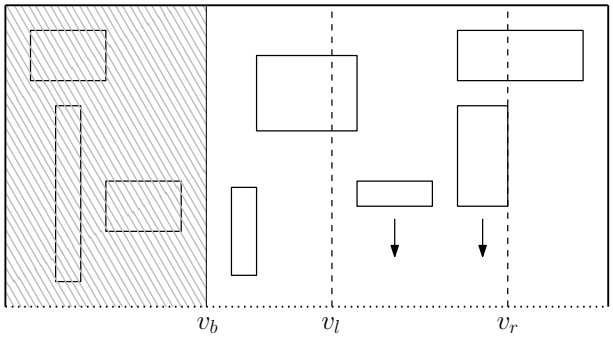


Figure 5: Illustrating Problem 4.

Problem 4 (No-Left-Escape)**.** *Given integers* $0 \leq i \leq n$ *and* $1 \leq b, l, r \leq k$, No-Left-Escape(i, b, l, r)*, is defined as the maximum number of rectangles among* $R_1, \ldots, R_i$ *which can be routed in unit density under the following restrictions:*

- *only rectangles to the right of* v_b *are allowed to escape,*
- *no rectangle is allowed to escape leftward, and*
- *only rectangle between* v_l *and* v_r *are allowed to escape downward.*

See Figure 5 for an illustration. To find the value of No-Left-Escape(i, b, l, r) recursively, we consider all possible actions for R_i. The first possible action for R_i is not to escape at all. In this case, the solution is equal to the solution of No-Left-Escape$(i-1, b, l, r)$. The other possible three actions for R_i are listed below. In what follows, we assume that the considered direction is *free* for R_i, and that R_i is allowed to escape through that direction according to the problem restrictions described above. Otherwise, we simply rule out that direction from the possible actions of R_i. Let v_α and v_β be the vertical lines obtained by extending the left and the right sides of R_i, respectively.

- *Downward* If R_i escapes downward, the maximum number of rectangles among $R_1, \ldots, R_{i-1}$ that can escape is equal to No-Left-Escape$(i-1, b, l, r)$, since routing R_i imposes no new restriction on $R_1, \ldots, R_{i-1}$.
- *Upward* If R_i escapes upward, one additional restriction must be considered: rectangles not to the right of v_β cannot escape rightward. Therefore, by the problem definition, each rectangle between v_b and v_β can only escape upward or downward. As such, escaping the maximum number of rectangles between v_b and v_β can be solved independently using subroutines One-Direction and Two-Directions, depending on the position of v_l and v_r. The rectangles to the right of v_β form another subproblem, whose optimal answer is No-Left-Escape$(i-1, \beta, l, r)$.
- *Rightward* By escaping rightward, one more restriction is posed to other rectangles: for any $1 \leq j < i$, R_j can escape downward if its initial place is not only to the left of v_r, but is also to the left of v_α. It means that if initial position of R_j is not to the left of $v_{\min\{r,\alpha\}}$, it cannot be routed downward. Therefore, the optimum answer for $R_1, \ldots, R_{i-1}$ in this case is No-Left-Escape$(i-1, b, l, \min\{r, \alpha\})$.

Algorithm 1 MAX-ROUTE(i, l, r)

if $i = 0$ **then**
  return 0
$ans_n \leftarrow$ MAX-ROUTE$(i-1, l, r)$
$ans_d \leftarrow ans_u \leftarrow ans_l \leftarrow ans_r \leftarrow 0$
$\alpha, \beta \leftarrow$ indices of the vertical lines through the left and the right sides of R_i, respectively.
if *down* is feasible for R_i **then**
  $ans_d \leftarrow$ MAX-ROUTE$(i-1, l, r) + 1$
if *left* is feasible for R_i **then**
  $ans_l \leftarrow$ MAX-ROUTE$(i-1, \max\{l, \beta\}, r) + 1$
if *right* is feasible for R_i **then**
  $ans_r \leftarrow$ MAX-ROUTE$(i-1, l, \min\{r, \alpha\}) + 1$
if *up* is feasible for R_i **then**
  $ans_u \leftarrow$ NO-RIGHT-ESCAPE$(i-1, \alpha, l, r)$ + NO-LEFT-ESCAPE$(i-1, \beta, l, r) + 1$
return $\max\{ans_n, ans_d, ans_u, ans_l, ans_r\}$

The *No-Right-Escape* is analogously defined, and can be solved similarly. Now, we have all ingredients necessary to solve Problem 3. Indeed, we solve the following more general problem:

Problem 5 (Max-Route)**.** *Given integers $0 \le i \le n$ and $1 \le l, r \le k$, find the maximum number of rectangles among $R_1, \ldots, R_i$ that can be routed in unit density under the following restriction: if a rectangle is not between v_l and v_r, it is not allowed to escape downward.*

The procedure MAX-ROUTE(i, l, r) defined in Algorithm 1 solves the problem as follows. We consider all possible actions for R_i. Except for escaping upward, all remaining actions can be solved like the previous problems. When R_i escapes upward, it is enough to calculate the sum of NO-LEFT-ESCAPE$(i-1, \beta, r, l)$ and NO-RIGHT-ESCAPE$(i-1, \alpha, r, l)$, since routing rectangles to the left of v_α and routing rectangles to the right of v_β are two independent subproblems.

Lemma 1. *Problem 5 can be solved in $O(n^4)$ time.*

Proof. To solve this problem, consider a dynamic-programming version of MAX-ROUTE algorithm. First, using a greedy algorithm, solve the ONE-DIRECTION and TWO-DIRECTIONS problems for any tuple (i, l, r), and store them in a table. This can be done in $O(n^4)$ time. Then, by the definition of problem 4, we can solve NO-LEFT-ESCAPE and NO-RIGHT-ESCAPE independently using dynamic programming. Note that in dynamic programming, the value of each tuple (i, b, l, r) can be obtained in $O(1)$ time from four previously-calculated values as described above. Putting all together, by using the description of Problem 5, each value of MAX-ROUTE(i, l, r) can be obtained from the previously-calculated values of this function, or solutions of NO-LEFT-ESCAPE and NO-RIGHT-ESCAPE. This can be done in

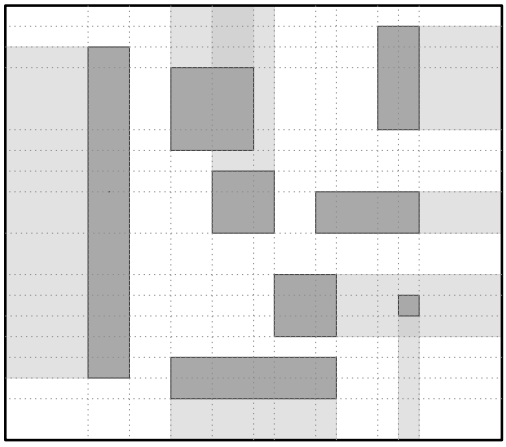


Figure 6: The grid cells for an instance of the rectangle escape problem.

$O(1)$ time assuming that the previous values are stored in a table. Thus, using a dynamic programming algorithm, Problem 5 can be solved in $O(n^4)$ time and space. □

The following theorem summarizes the result of this section.

Theorem 3. *1-REP can be solved in $O(n^4)$ time.*

Proof. Observe that the answer to 1-REP is *yes* iff the answer to Problem 5 for $(n, 1, k)$ is equal to n, where k is the index of the rightmost vertical line. The running time therefore follows from Lemma 1. □

4 A Randomized Approximation Algorithm

As noted in Section 2, the rectangle escape problem is NP-hard, even when the optimal density is 2. Therefore, it is natural to look for approximation algorithms for the problem. The current best approximation algorithm is due to Ma and Wong [5], which achieves an approximation factor of 4. The algorithm is based on a deterministic rounding of an integer programming formulation of the problem. In this section, we show that a standard randomized rounding technique [8] applied to the same integer programming formulation of the problem, yields an approximation factor of $1 + \varepsilon$, when the optimal density is at least $c_\varepsilon \log n$, for some constant c_ε.

The integer programming formulation of the problem is as follows. Let $S = \{r_1, \ldots, r_n\}$ be the set of input rectangles inside a region R. We build a grid on top of R by extending each side of the rectangles in S into a line (see Figure 6). This partitions R into a set $\mathcal{C}$ of $O(n^2)$ grid cells, where the density over each cell is fixed.

For each rectangle r_i, we define four 0-1 variables $x_{i,l}, x_{i,r}, x_{i,u}$, and $x_{i,d}$, corresponding to the four directions left, right, up, and down, respectively. For a direction $\lambda \in \{l, r, u, d\}$, we set $x_{i,\lambda} = 1$ if r_i is escaped toward direction λ, otherwise, $x_{i,\lambda} = 0$. Since any rectangle r_i can escape toward only one direction, we have the constraint $x_{i,l} + x_{i,r} + x_{i,u} + x_{i,d} = 1$. For each grid cell $c \in \mathcal{C}$, let $P_c = \{(i, \lambda) \mid r_i \text{ passes } c \text{ if it goes toward direction } \lambda\}$. Note that if cell c is contained in r_i, then $(i, \lambda) \in P_c$ for all directions λ. Let Z be the maximum density over

Algorithm 2 RANDOMIZED-ROUNDING

1: find an optimal solution x^* to the LP relaxation (1)
2: route each r_i to direction λ according to the probability distribution $x^*_{i,\lambda}$

the region R. Then, for each grid cell $c \in \mathcal{C}$ we can add the constraint $\sum_{(i,\lambda)\in P_c} x_{i,\lambda} \leq Z$. Now, the problem can be formulated as the following integer program.

$$\begin{aligned} &\text{minimize} && Z && && (1)\\ &\text{subject to} && \sum_{(i,\lambda)\in P_c} x_{i,\lambda} \leq Z && \forall c \in \mathcal{C} \\ & && x_{i,l} + x_{i,r} + x_{i,u} + x_{i,d} = 1 && \forall\ 1 \leq i \leq n \\ & && x_{i,l}, x_{i,r}, x_{i,u}, x_{i,d} \in \{0,1\} && \forall\ 1 \leq i \leq n \end{aligned}$$

The randomized rounding algorithm for the rectangle escape problem is provided in Algorithm 2. The algorithm works as follows. We first relax the integer program to a linear program by replacing the constraints $x_{i,\lambda} \in \{0,1\}$ with $0 \leq x_{i,\lambda} \leq 1$, and solve the linear programming relaxation to obtain a solution x^* with objective value Z^*. Then, we randomly route each rectangle to exactly one direction by interpreting the value of $x^*_{i,\lambda}$ as the probability of routing r_i toward direction λ.

Theorem 4. *Algorithm 2 is a $(1+\varepsilon)$-approximation algorithm for the rectangle escape problem with high probability, when $Z^* \geq 9/\varepsilon^2 \ln n$.*

Proof. For each cell c, let D_c be the density of c in the solution returned by the algorithm. Define the random variables $X_{i,\lambda}$, where $X_{i,\lambda} = 1$ if rectangle r_i is routed toward direction λ by the algorithm, and $X_{i,\lambda} = 0$ otherwise. Then, we have $D_c = \sum_{(i,\lambda)\in P_c} X_{i,\lambda}$. Therefore,

$$\begin{aligned} E[D_c] &= \sum_{(i,\lambda)\in P_c} E[X_{i,\lambda}] \\ &= \sum_{(i,\lambda)\in P_c} \Pr\{X_{i,\lambda} = 1\} \\ &= \sum_{(i,\lambda)\in P_c} x^*_{i,\lambda} \qquad \text{(by line 2 of algorithm)} \\ &\leq Z^*. \qquad \text{(by LP constraint)} \end{aligned}$$

Moreover, for each cell c, the variables $X_{i,\lambda}$ for all $(i,\lambda) \in P_c$ are independent. To see this, notice that there are two types of variables contributing to the density of c. If c is contained in a rectangle r_i, then $X_{i,\lambda}$, for all directions λ, pass through c. In this case, we can replace these four variables in the constraint of c by just a number 1, since one and exactly one of these variables will be 1 in any optimal solution of LP. If c is not contained in r_i, then (i,λ) contributes to the density of c for at most one value of λ, since no two directions of r_i can pass through c simultaneously. Therefore, after substituting the first type of variables in the

constraint of cell c by 1, all other variables $X_{i,\lambda}$ for all $(i,\lambda) \in P_c$ are independent, due to the fact that the direction of rectangles are chosen independently.

We can now use Chernoff bound to show that D_c is close to Z^* with high probability. We use the following statement of Chernoff bound: If $X_1, \ldots, X_n$ are independent 0-1 random variables, $X = \sum X_i$, $E[X] \leq U$, and $0 \leq \varepsilon \leq 1$, then $\Pr\{X \geq (1+\varepsilon)U\} \leq e^{-U\varepsilon^2/3}$. Since $E[D_c] \leq Z^*$, by Chernoff bound we have

$$\Pr\{D_c \geq (1+\varepsilon)Z^*\} \leq e^{-Z^*\varepsilon^2/3}.$$

The solution produced by our algorithm has density $\max_c\{D_c\}$. Since there are at most $(2n)^2$ grid cells, assuming $Z^* \geq c_\varepsilon \ln n$ for some constant $c_\varepsilon > 0$, we get

$$\begin{aligned}
\Pr\{\max_c\{D_c\} \geq (1+\varepsilon)Z^*\} &\leq \sum_c \Pr\{D_c \geq (1+\varepsilon)Z^*\} \\
&\leq (2n)^2 \times n^{-c_\varepsilon\varepsilon^2/3} \\
&= 4n^{2-(c_\varepsilon\varepsilon^2/3)}.
\end{aligned}$$

Therefore, for a proper constant $c_\varepsilon \geq 9/\varepsilon^2$, the probability that the solution returned by our algorithm is greater than $(1+\varepsilon)Z^*$ is at most $\frac{4}{n}$. Taking into account that $Z^* \leq \mathrm{OPT}$, it shows that our algorithm has an approximation factor of $1+\varepsilon$ with high probability if $Z^* \geq c_\varepsilon \ln n$. □

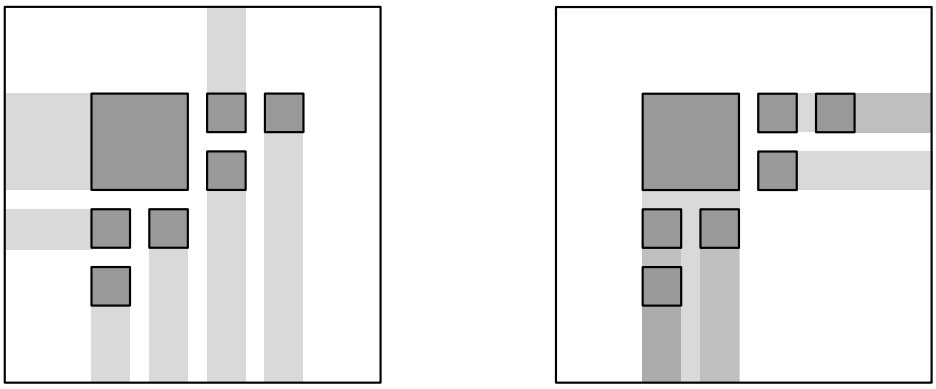
5 Bidirectional Rectangle Escape

There are applications in which some of the directions may not be available for routing the components. For these applications, it is natural to consider a restricted version of the REP problem, in which rectangles are allowed to escape only toward a fixed subset of directions. In particular, we consider the *bidirectional REP* problem, in which rectangles can only escape toward two adjacent sides of R. We assume, w.o.l.g., that these two directions are right and down. A motivation for considering such restricted instances is their potential usage in approximating the REP, as demonstrated in the following lemma.

Lemma 2. *Any algorithm for bidirectional REP yields a 3-approximation for the REP problem.*

Proof. Let $\mathcal{A}$ be an exact algorithm for the bidirectional REP problem. Given an instance S of the REP problem, we run $\mathcal{A}$ on S and return the solution. We show that the output of $\mathcal{A}$ is a 3-approximation. For a set $T \subseteq S$, we denote by $\mathrm{dens}(T)$ the minimum density by which we can route the rectangles in T toward two adjacent borders, say, right and bottom.

Consider an optimal solution for the REP problem on the set S, with density $\mathtt{opt}(S)$. Let S_ℓ, S_r, S_u, and S_d denote the subset of rectangles in S escaped in the optimal solution toward left, right, up, and down, respectively. We construct a bidirectional solution by routing $S_r \cup S_\ell$ toward right, and routing $S_d \cup S_u$ toward down. The density of this solution is at most $\mathrm{dens}(S_r \cup S_d) + \mathrm{dens}(S_u) + \mathrm{dens}(S_\ell) \leq 3 \cdot \mathtt{opt}(S)$, showing that the approximation factor of $\mathcal{A}$ is at most 3. To see that this approximation factor is tight, consider a REP instance illustrated in Figure 7. The optimal density for this instance is 1, while its corresponding bidirectional REP instance has density 3. □



(a) A REP instance (b) Corresponding BREP instance

Figure 7: A tight example for approximating REP using bidirectional REP.

The decision version of bidirectional REP, denoted by bidirectional k-REP, is defined analogous to the k-REP problem. In the following, we show that the bidirectional k-REP problem, despite being restricted to only two adjacent directions, is still NP-complete.

Theorem 5. *The bidirectional k-REP problem is NP-complete for any $k \geq 3$.*

Proof. Given an instance of 3-SAT, we construct an instance of bidirectional 3-REP similar to the construction used in the proof of Theorem 1 with the following differences. The top and the left regions are not required any more and are omitted. The long rectangle at the bottom of clauses region, as well as the long rectangle at the right side of variables region are both duplicated (see Figure 8). The long rectangle at the left side of clauses region is also omitted, as no rectangle can escape to the left. Now, it is easy to verify that a similar argument stated in the proof of Theorem 1 holds here: the answer to the constructed bidirectional 3-REP is yes if and only if the given 3-SAT instance is satisfiable.

For $k > 3$, we construct an instance of bidirectional k-REP using a similar recursive construction described in Figure 4. We only omit the left and the top instances of R_{k-1}, because the middle rectangle Q can only escape toward right or down. □

As a corollary of Theorem 5, we obtain the following inapproximability result.

Theorem 6. *The bidirectional REP problem admits no α-approximation for any $\alpha < 4/3$, unless $P = NP$.*

Proof. Suppose by way of contradiction that there is an α-approximation algorithm with $\alpha < 4/3$ for the bidirectional REP problem. If we run this algorithm on an instance with optimum density 3, it returns a solution with density less than $(4/3) \times 3$, which has to be 3. Thus, we can use such an algorithm to solve bidirectional 3-REP exactly in polynomial time, contradicting the NP-completeness of the problem for $k = 3$. □

Since bidirectional REP is NP-complete, it is natural to look for approximation algorithms for it. The following theorem provides a factor-2 approximation.

Theorem 7. *There is a 2-approximation algorithm for the bidirectional REP problem.*

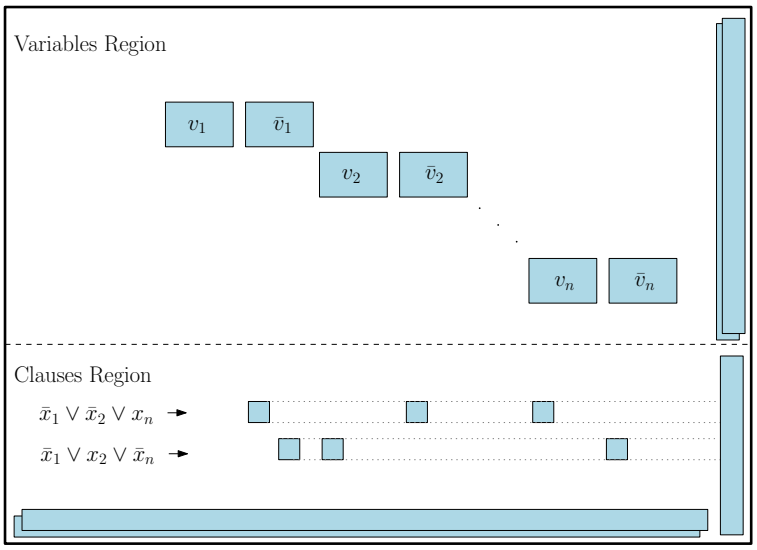


Figure 8: Reduction from 3-SAT to bidirectional 3-REP.

Proof. We write a linear program similar to what we used for the REP problem. For each rectangle r_i, we define two 0-1 variables $x_{i,r}$ and $x_{i,d}$ specifying whether r_i escapes to the right or down, respectively. Moreover, let $\mathcal{C}$ and P_c be the same sets defined in Section 4. Now, the bidirectional REP problem can be formulated as the following integer program.

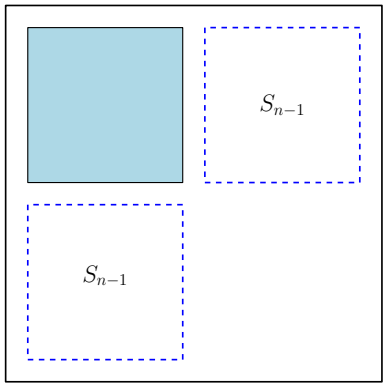
$$\begin{aligned} &\text{minimize} && Z && && (2)\\ &\text{subject to} && \sum_{(i,\lambda)\in P_c} x_{i,\lambda} \leq Z && \forall c \in \mathcal{C} \\ & && x_{i,r} + x_{i,d} = 1 && \forall\ 1 \leq i \leq n \\ & && x_{i,r}, x_{i,d} \in \{0,1\} && \forall\ 1 \leq i \leq n \end{aligned}$$

We can now use a deterministic rounding technique as follows. We first relax the integer program (2) to a linear program by letting each 0/1 variable to get a real value between 0 and 1. We then solve the linear programming relaxation to obtain a solution x^*, yielding an objective value Z^*. Now, for each rectangle r_i, if $x^*_{i,r} > x^*_{i,d}$, we set $x_{i,r} = 1$, meaning that r_i escapes toward right, otherwise, we set $x_{i,d} = 1$ to escape r_i toward down. Since one of the two variables corresponding to each rectangle has value at least $1/2$, the obtained solution has a density of at most $2Z^*$, and hence, it is a 2-approximation. □

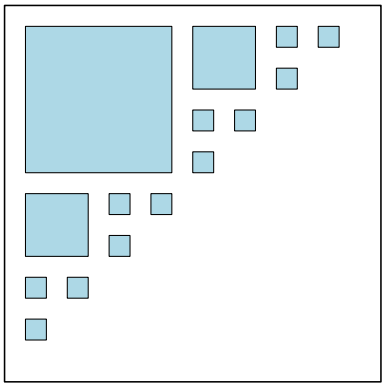
We demonstrate a class of instances for the bidirectional REP problem, that shows a lower bound of 1.77 on the integrality gap of the linear program defined above. This implies that any approximation algorithm which is based on the relaxation of the linear program (2) cannot yield an approximation factor better than 1.77.

Consider a structure S_n which is recursively defined as follows. Each S_n is an instance of the bidirectional REP problem whose optimal solution has value n. S_1 is just a single square, and its optimal solution is obviously 1. S_n is composed of a big square and two instances of S_{n-1} which are put to the right and bottom of this square, as shown in Figure 9.

The following lemma shows a lower bound of 2 on the min-max density of points in S_n.



(a) The recursive structure of S_n



(b) An illustration for S_4

Figure 9: The proposed structure with a large integrality gap.

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
ILP solution	1	2	3	4	5	6	7	8	9
LP solution	1	1.5	2	2.5	3	3.517	4.035	4.556	5.07
Integrality gap	1	1.33	1.5	1.6	1.67	1.70	1.73	1.75	1.77

Table 1: The integrality gap for S_n

Lemma 3. *The min-max density of points in S_n is at least n.*

Proof. We prove this by induction on n. For the base case, when $n = 1$, the statement is clearly true. Now, suppose that the min-max density for S_{n-1} is at least $n-1$ ($n > 1$). Then, there is no way for the big square in S_n (see Figure 9a) to escape other than passing through a point with density at least $n - 1$. Therefore, the min-max density of S_n is at least n. □

By letting all rectangles escape downward, we obtain a solution with maximum density of n. Therefore, the integral solution of S_n is exactly n. The optimal solution for the bidirected version of LP (2) is computed using a computer program, and the results are reported in Table 1. The results prove a lower bound of 1.77 on the integrality gap. A larger integrality gap can be obtained by increasing n, however, it does not seem to reach the upper bound 2.

6 Conclusions

In this paper, we provided some new insights into the rectangle escape problem. In particular, we presented a lower bound of 3/2 on the approximability of the rectangle escape problem, and a $(1 + \varepsilon)$-approximation algorithm for the problem when the optimal density is sufficiently large. Several intriguing questions remain open. The main question is whether an approximation factor better than 4 is possible for the problem in general case. Finding approximation factors better than 2 (resp., 3) for the restricted 2-sided (resp., 3-sided) version of the problem is also interesting. The complexity of bidirectional k-REP for $k = 2$, as well as the complexity of bidirectional k-REP when the escape directions are opposite are also open.

References

[1] A. Ahmadinejad and H. Zarrabi-Zadeh. The maximum disjoint set of boundary rectangles. In *Proceedings of the 26th Canadian Conference on Computational Geometry*, CCCG '14, pages 302–307, 2014.

[2] S. Assadi, E. Emamjomeh-Zadeh, S. Yazdanbod, and H. Zarrabi-Zadeh. On the rectangle escape problem. In *Proceedings of the 25th Canadian Conference on Computational Geometry*, CCCG '13, pages 235–240, 2013.

[3] H. Kong, Q. Ma, T. Yan, and M. D. F. Wong. An optimal algorithm for finding disjoint rectangles and its application to PCB routing. In *Proceedings of the 47th ACM/EDAC/IEEE Design Automation Conference*, DAC '10, pages 212–217, 2010.

[4] H. Kong, T. Yan, and M. D. F. Wong. Automatic bus planner for dense PCBs. In *Proceedings of the 46th ACM/EDAC/IEEE Design Automation Conference*, DAC '09, pages 326–331, 2009.

[5] Q. Ma and M. D. F. Wong. NP-completeness and an approximation algorithm for rectangle escape problem with application to PCB routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(9):1356–1365, 2012.

[6] Q. Ma, E. Young, and M. D. F. Wong. An optimal algorithm for layer assignment of bus escape routing on PCBs. In *Proceedings of the 48th ACM/EDAC/IEEE Design Automation Conference*, pages 176–181, 2011.

[7] M. M. Ozdal, M. D. F. Wong, and P. S. Honsinger. An escape routing framework for dense boards with high-speed design constraints. In *Proceedings of the 2005 IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '05, pages 759–766, 2005.

[8] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[9] P.-C. Wu, Q. Ma, and M. D. Wong. An ILP-based automatic bus planner for dense PCBs. In *Proceedings of the 18th Asia South Pacific Design Automation Conference*, ASPDAC '13, pages 181–186, 2013.

[10] J. T. Yan and Z. W. Chen. Direction-constrained layer assignment for rectangle escape routing. In *Proceedings of the 2012 IEEE International System-on-Chip Conference*, SOCC '12, pages 254–259, 2012.

[11] J. T. Yan, J. M. Chung, and Z. W. Chen. Density-reduction-oriented layer assignment for rectangle escape routing. In *Proceedings of the Great Lakes Symposium on VLSI*, GLSVLSI '12, pages 275–278, 2012.

[12] T. Yan, H. Kong, and M. D. F. Wong. Optimal layer assignment for escape routing of buses. In *Proceedings of the 2009 IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '09, pages 245–248, 2009.