On the Rectangle Escape Problem

Sepehr Assadi* Ehsan Emamjomeh-Zadeh* Sadra Yazdanbod* Hamid Zarrabi-Zadeh*†

Abstract

Motivated by a bus routing application, we study the following *rectangle escape* problem: Given a set S of n rectangles inside a 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, where 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 upon the current best 4-approximation algorithm available for the problem. When the optimal density is one, we provide an exact algorithm that finds an optimal solution in $O(n^4)$ time, improving upon the current best $O(n^6)$-time algorithm.

1 Introduction

Consider a set of electrical components (e.g., chips) placed on a printed circuit board (PCB), where both the board and the chips are axis-parallel rectangles. We want to connect each chip to one of the four sides of the board using a rectangular bus (see Figure 1). The goal is to find a routing direction for the chips so that the maximum number of bus conflicts at any single point over the board is minimized. This is equivalent to minimizing the number of layers needed for routing all the chips on the board. The problem is called the *rectangle escape problem* [3], and has been extensively studied in the literature (see, e.g., [1, 2, 3, 4, 5, 7, 8, 9, 10]). The problem is formally defined as follows:

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.*

*Department of Computer Engineering, Sharif University of Technology. {s_asadi,emamjomeh,yazdanbod}@ce.sharif.edu, zarrabi@sharif.edu

†School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran.

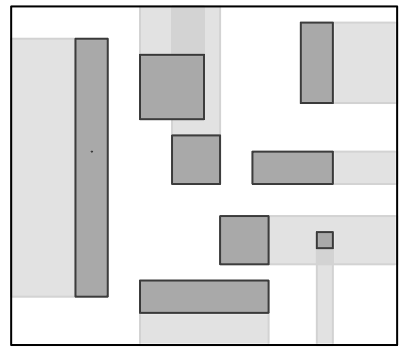

Figure 1: An instance of the rectangle escape problem. Chips are shown in dark, and buses in light gray.

An example of the rectangle escape problem is illustrated in Figure 1. In this example, the optimal density, which is equal to the minimum number of layers needed for routing the chips is two.

The rectangle escape problem is known to be NP-hard [3]. The *decision version* of the problem, called k-REP, is defined as follows: Given an instance of the rectangle escape problem and an integer $k \geqslant 1$, 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 $k = 3$ [3]. The best current approximation algorithm for the optimization version of the problem is due to Ma *et al.* [3] 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 a polynomial-time algorithm for the related *maximum disjoint subset* problem, for which an $O(n^6)$-time algorithm is proposed by Kong *et al.* [1].

Our results. In this paper, we obtain some new 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 $k \geqslant 2$. 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.

- We present a new algorithm that solves the 1-REP problem in $O(n^4)$ time, improving upon the current best solution for the problem that requires $O(n^6)$ time [1]. Our algorithm can indeed solve the following more general optimization version of the problem: given an instance of the rectangle escape problem, find a maximum-size subset of rectangles in S that can be routed disjointly.

- 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 *et al.* [3] 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.

2 Hardness Result

We first show that the k-REP problem is NP-complete, for any $k \geqslant 2$. 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 $k \geqslant 2$, even if all input rectangles are disjoint.*

Proof. We prove by reduction from 3-SAT. The reduction is similar to that of [3], but uses a more clever construction to handle the special case of $k = 2$, and 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 2. 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 2. 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. Again, no two literal rectangles intersect, and no two of them can be stabbed by a vertical line.

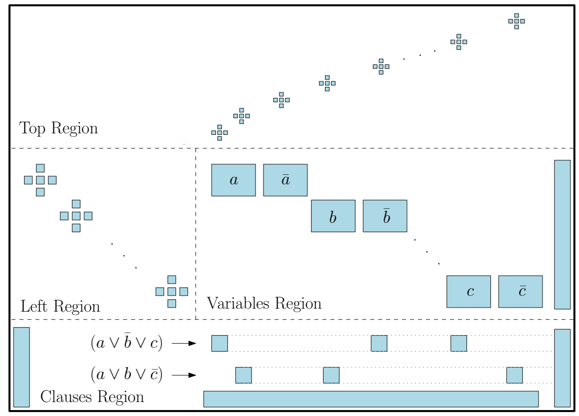

Figure 2: Reduction from 3-SAT to 2-REP.

- 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 2. 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

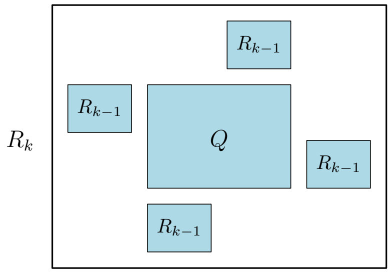

Figure 3: Constructing an instance of k-REP from four instances of $(k-1)$-REP.

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 $k = 2$.

To show NP-completeness for other values of $k > 2$, we use the following 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 3. 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. The opposite side can be proved analogously (details are omitted in this version). □

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 in $O(n^4)$ time, improving upon the previous solution due to Kong *et al.* [1] that requires $O(n^6)$ time. Our algorithm solves the following optimization problem.

Problem 2 (Maximum Disjoint Routing) *Given an instance of the rectangle escape problem (Problem 1) 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 2 can also solve 1-REP: we first find the maximum number of rectangles that can be routed disjointly, and then verify 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 $\{v_1, \ldots, v_k\}$ $(k \leqslant 2n)$ as the set of all vertical lines obtained by extending the vertical sides of the rectangles, sorted from left to right.

To solve Problem 2, we first solve two simpler cases in which the escaping directions are only vertical. Given integers $0 \leqslant i \leqslant n$ and $1 \leqslant l, r \leqslant k$, 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.

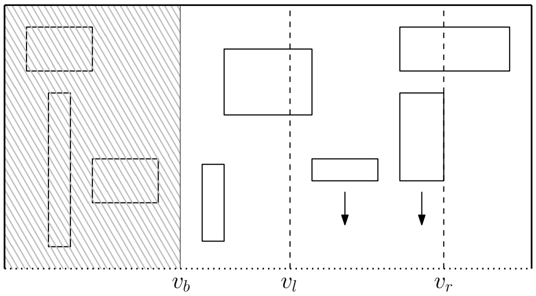

Figure 4: Illustrating Problem 3.

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 $(1 \leqslant j \leqslant i)$ 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.

Problem 3 (No-Left-Escape) *Given integers* $0 \leqslant i \leqslant n$ *and* $1 \leqslant b, l, r \leqslant k$*,* 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 4 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 $R_1, \ldots, R_{i-1}$.

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 $\max\{ans_n, ans_d, ans_u, ans_l, ans_r\}$

- *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 $1 \leqslant j < i$, 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$(i-1, b, l, \min\{r, \alpha\})$.

The *No-Right-Escape* is analogously defined, and can be solved similarly. Now, we have all ingredients necessary to solve Problem 2. Indeed, we solve the following more general problem:

Problem 4 (Max-Route) *Given integers* $0 \leqslant i \leqslant n$ *and* $1 \leqslant l, r \leqslant k$*, 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 3 *Problem 4 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 3, 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 4, 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 $O(1)$ time assuming that the previous values are stored in a table. Thus, using a dynamic programming algorithm, Problem 4 can be solved in $O(n^4)$ time and space. □

The following theorem summarizes the result of this section.

Theorem 4 *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 4 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 3. □

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 *et al.* [3], 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 [6] 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 5). This partitions R into a set $\mathcal{C}$ of $O(n^2)$ grid cells, where the density over each cell is fixed.

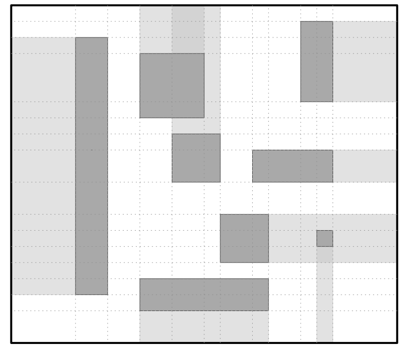

Figure 5: The grid cells for an instance of the rectangle escape problem.

Algorithm 2 Randomized-Rounding

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

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

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

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 $x_{i,\lambda} \geqslant 0$, 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 5 *Algorithm 2 is a $(1+\varepsilon)$-approximation algorithm for the rectangle escape problem with high probability, when $Z^* \geqslant 9/\varepsilon^2 \ln n$.*

Proof. For each cell c, let D_c be the density of c in the solution returned by the algorithm. Define 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,

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

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, $X = \sum X_i$, $E[X] \leqslant U$, and $0 \leqslant \varepsilon \leqslant 1$, then $\Pr\{X \geqslant (1+\varepsilon)U\} \leqslant e^{-U\varepsilon^2/3}$. Since $E[D_c] \leqslant Z^*$, by Chernoff bound we have

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

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

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

Therefore, for a proper constant $c_\varepsilon \geqslant 9/\varepsilon^2$, 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 $Z^* \leqslant \mathrm{OPT}$, it shows that our algorithm has an approximation factor of $1+\varepsilon$ with high probability if $Z^* \geqslant c_\varepsilon \ln n$. □

5 Conclusions

In this paper, we presented some new results on the rectangle escape problem. In particular, we presented a lower bound of 3/2 on the approximability of the problem, and a $(1+\varepsilon)$-approximation algorithm for the problem when the optimal density is high enough. It remains open what the best approximation factor is for the problem in general case.

Acknowledgments The authors would like to thank Hesam Monfared for suggesting the rectangle escape problem, and the anonymous referees for their helpful comments.

References

[1] 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 *Proc. 47th ACM/EDAC/IEEE Design Automation Conf.*, DAC '10, pages 212–217, 2010.

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

[3] Q. Ma, H. Kong, M. D. F. Wong, and E. F. Y. Young. A provably good approximation algorithm for rectangle escape problem with application to PCB routing. In *Proc. 16th Asia South Pacific Design Automation Conf.*, ASPDAC '11, pages 843–848, 2011.

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

[5] M. M. Ozdal, M. D. F. Wong, and P. S. Honsinger. An escape routing framework for dense boards with high-speed design constraints. In *Proc. 2005 IEEE/ACM Internat. Conf. Computer-Aided Design*, ICCAD '05, pages 759–766, 2005.

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

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

[8] J. T. Yan and Z. W. Chen. Direction-constrained layer assignment for rectangle escape routing. In *Proc. 2012 IEEE Internat. System-on-Chip Conf.*, SOCC '12, pages 254–259, 2012.

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

[10] T. Yan, H. Kong, and M. D. F. Wong. Optimal layer assignment for escape routing of buses. In *Proc. 2009 IEEE/ACM Internat. Conf. Computer-Aided Design*, ICCAD '09, pages 245–248, 2009.