A Provably Good Approximation Algorithm for Rectangle Escape Problem with Application to PCB Routing

Qiang Ma[†] Hui Kong[†] Martin D. F. Wong[†] Evangeline F. Y. Young[‡]

[†]Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign
[‡]Department of Computer Science and Engineering, The Chinese University of Hong Kong
Email: qiangma1@illinois.edu, huikong2@illinois.edu, mdfwong@illinois.edu, fyyoung@cse.cuhk.edu.hk

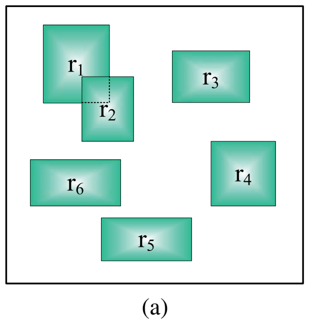
***Abstract*— In this paper, we introduce and study the Rectangle Escape Problem (REP), which is motivated by PCB bus escape routing. Given a rectangular region *R* and a set *S* of rectangles within *R*, the REP is to choose a direction for each rectangle to escape to the boundary of *R*, such that the resultant maximum density over *R* is minimized. We prove that the REP is NP-Complete, and show that it can be formulated as an Integer Linear Program (ILP). A provably good approximation algorithm for the REP is developed by applying Linear Programming (LP) relaxation and a special rounding technique to the ILP. This approximation algorithm is also shown to work for a more general version of REP with weights (weighted REP). In addition, an iterative refinement procedure is proposed as a postprocessing step to further improve the results. Our approach is tested on a set of industrial PCB bus escape routing problems. Experimental results show that the optimal solution can be obtained within 3 seconds for each of the test cases.**

I. Introduction

In the Rectangle Escape Problem (REP), we are given a rectangular region *R* and a set *S* of rectangles staying within *R*, where each rectangle has to "escape" to one of the four boundaries of *R*. By "escape" we mean extending the rectangle in a certain direction, namely, left, right, top or bottom, until it reaches the corresponding boundary of the rectangular region *R*. The objective of REP is to determine an escape direction for each rectangle in *S*, such that the resultant maximum density over the region *R* is minimized. In the region *R*, the density at a point is the number of rectangles containing this point, and the point with largest density defines the maximum density over *R*. Fig.1 illustrates the Rectangle Escape Problem: Fig.1 (a) shows the input rectangles in *S*, while Fig.1 (b) and (c) give two escape solutions to this problem. The shaded region attached to each rectangle is the extension of the corresponding rectangle after escaping. It is easy to see that the resultant maximum densities of the two escape solutions in Fig.1 (b) and (c) are respectively 2 and 3. The objective of the REP is to minimize the maximum density over the rectangular region *R*, so the solution in Fig.1 (b) is better than the solution in Fig.1 (c).

The study of the REP is motivated by the escape routing problem on the bus level in PCBs. In the past few years, as the dimensions of packages and PCBs keep decreasing and the pin counts and routing layers keep increasing, the escape routing problem, which is to route nets from their pins to the component boundaries, becomes more and more critical [3], [8]–[10]. In a PCB bus escape routing instance, the nets of a bus are preferred to be routed together, without mixing with the nets from other buses [4]–[6]. From industrial manual routing solutions, we observe that the escape routes of all the nets of a bus are typically within one of its projection rectangles, which can be obtained by extending the bounding box of the pin cluster of the bus to one of the component boundaries. Fig.2 shows the four projection rectangles of a bus, together with an example escape routing to the right boundary. The bounding box of the pin cluster of a bus can be represented by a rectangle in REP, while the four projection rectangles correspond to the escapes in four directions.

This work was partially supported by the National Science Foundation under grant CCF-1017516.



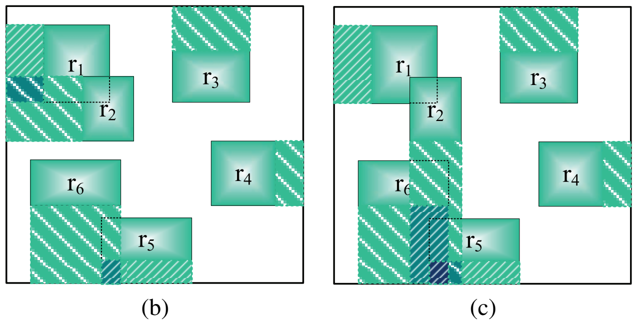


Fig. 1. Illustration of Rectangle Escape Problem: (a) input rectangles; (b) an escape solution with maximum density equal to 2; (c) an escape solution with maximum density equal to 3.

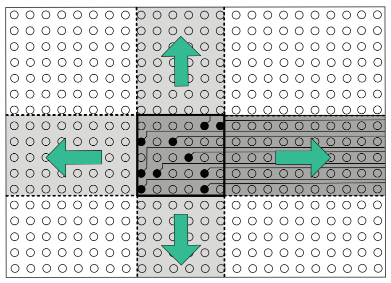


Fig. 2. The four projection rectangles of a bus and its escape routing to the right boundary of the component.

When the escape routes of two buses conflict, they have to be routed on different layers. Fig.3 demonstrates two types of conflict. The fabrication cost dramatically increases when more layers are needed, so we want to use as few layers as possible to accommodate all the buses. According to our experience, the maximum density is usually a good indicator of the number of layers needed (although

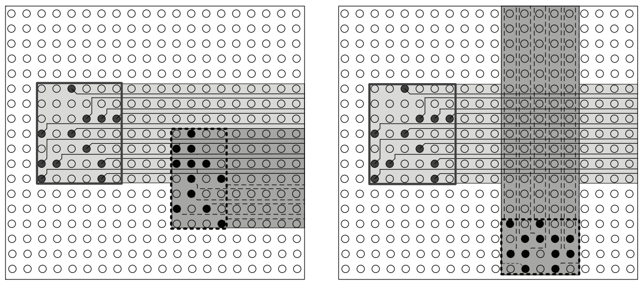

(a) (b)

Fig. 3. The routes of two buses conflict.

theoretically it is only a lower bound of the necessary number of layers). Therefore, in our Rectangle Escape Problem, we try to minimize the maximum density. Note that the actual layer assignment will have to be deferred until bus planning between various components is done so that a more global view of all bus intersections is available [5].

Our contributions in this paper can be summarized as follows:

- We prove that the REP is NP-Complete.
- A 4-approximation algorithm for REP is developed.
- This approximation algorithm is also shown to work for a more general version of REP with weights (weighted REP).
- Our algorithm is tested on a set of industrial PCB bus escape routing cases, and results show that the optimal solution can be obtained within 3 seconds for each test case, which confirms the effectiveness and efficiency of our approach.

The remainder of this paper is organized as follows. The REP is defined in Section II. We will prove the NP-Completeness of the REP in Section III, and present our approximation algorithm for the REP in Section IV. Experimental results are reported in Section V before concluding in the last section.

II. Problem Definition

An REP instance $\mathcal{R}$ contains a rectangular region R and a set S of n rectangles $\{r_1, r_2, \ldots, r_n\}$. Given a candidate escape solution for the instance $\mathcal{R}$, let d_{max} denote the resultant maximum density over the rectangular region R. Now the REP can be defined as follows:

RECTANGLE ESCAPE PROBLEM (REP)
INSTANCE: A rectangular region R and a set S of n rectangles $\{r_1, r_2, \ldots, r_n\}$ residing within R.

QUESTION: Each rectangle $r_i \in S$ chooses a direction to escape, such that d_{max} is minimized.

III. Proof of NP-Completeness

In this section, we show that REP is NP-Complete. In order to facilitate the proof of NP-Completeness, the decision version of REP is described as follows:

DECISION VERSION OF REP
INSTANCE: An integer k, a rectangular region R and a set S of n rectangles $\{r_1, r_2, \ldots, r_n\}$ residing within R.

QUESTION: Each rectangle $r_i \in S$ chooses a direction to escape, is there a solution such that $d_{max} \leq k$?

Lemma 1: REP is in NP.

Proof: Given an escape solution for the REP, where each rectangle r_i has its escape direction determined, we only need to show that the resultant maximum density d_{max} can be checked in polynomial time. Note that d_{max} equals the maximum clique size of the rectangle intersection graph of the n rectangles (after escaping). Lee showed in [7] that the maximum clique of a rectangle intersection graph can be computed in $O(n \log n)$ time. ■

We then prove REP is NP-Hard by using reduction from $3SAT$, which is a classical NP-Complete Problem [2].

3SAT
INSTANCE: A set U of n variables $\{x_1, x_2, \ldots, x_n\}$, a collection C of m clauses $\{c_1, c_2, \ldots, c_m\}$ over U such that $|c_i| = 3$, for $1 \leq i \leq m$.

QUESTION: Is there a satisfying assignment for C?

Given a $3SAT$ instance $\mathcal{S}$, we construct an REP instance $\mathcal{R}$, such that $\mathcal{S}$ has a satisfying truth assignment if and only if $\mathcal{R}$ has an escape solution with maximum density $d_{max} \leq 3$. The constructed REP instance $\mathcal{R}$ contains three types of rectangles, namely, blockage rectangles, variable rectangles and clause rectangles. The reduction is conducted as follows:

1) A rectangular region R is created at first. Three overlapping blockage rectangles are placed along the top boundary of R and another three overlapping blockage rectangles are placed along the left boundary of R, so that the top boundary and the left boundary are "blocked". The bottom boundary and the right boundary of R are referred to as the "True" boundary and the "False" boundary, respectively (see Fig.4).
2) The variable rectangles are placed along the diagonal of R. For each variable x_i, we create a rectangle x_i and a rectangle $\bar{x_i}$, and place them in such a way that the lower right corner of rectangle x_i overlaps with the upper left corner of rectangle $\bar{x_i}$. Note that the projection of variable rectangles for x_i and the projection of variable rectangles for x_j cannot overlap with each other (on either the "True" boundary or the "False" boundary), if $i \neq j$. In addition, for each variable x_i, we place two overlapping blockage rectangles along the "True" boundary, with their horizontal position as the projection of the two variable rectangles x_i and $\bar{x_i}$ on the "True" boundary; similarly, we place another two overlapping blockage rectangles along the "False" boundary, with their vertical position as the projection of the two variable rectangles x_i and $\bar{x_i}$ on the "False" boundary (see Fig.4 for illustration).
3) The clause rectangles are placed below all the variable rectangles. For each clause c_i, we create three copies of clause rectangle for c_i. If clause c_i contains x_j, we place one copy of clause rectangle c_i below the variable rectangle $\bar{x_j}$; if clause c_i contains $\bar{x_j}$, we place one copy of clause rectangle c_i below the variable rectangle x_j. Note that the clause rectangle c_i cannot be placed below the overlapping region of the two variable rectangles x_j and $\bar{x_j}$, and that the three copies of clause rectangle for c_i should be placed with the same vertical coordinate. In addition, a blockage rectangle for c_i needs to be placed along the "False" boundary, with its vertical position as the projection of the clause rectangles for c_i on the "False" boundary (see Fig.4 for illustration). Furthermore, we should make sure that the projection of a clause rectangle for c_i and the projection of a clause rectangle for c_j do not overlap (on either the "True" boundary or the "False" boundary), if $i \neq j$.

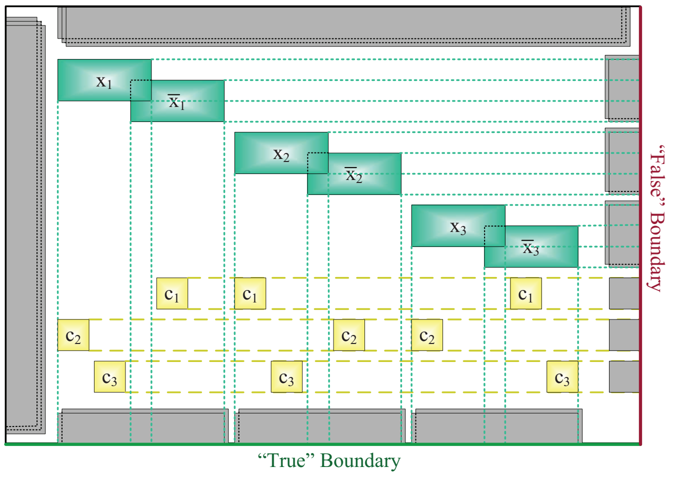


Fig. 4. The REP instance $\mathcal{R}$ constructed from the 3SAT instance $\mathcal{S}$ ($\mathcal{S}$ contains 3 variables $\{x_1,x_2,x_3\}$ and 3 clauses $\{c_1,c_2,c_3\}$, where $c_1 = (x_1 \vee \bar{x_2} \vee x_3)$, $c_2 = (\bar{x_1} \vee x_2 \vee \bar{x_3})$, and $c_3 = (\bar{x_1} \vee \bar{x_2} \vee x_3)$).

It is easy to see that the reduction can be done in polynomial time, with respect to the size of the 3SAT instance $\mathcal{S}$. Fig.4 shows the REP instance $\mathcal{R}$ constructed from a concrete 3SAT instance $\mathcal{S}$ with 3 variables $\{x_1,x_2,x_3\}$ and 3 clauses $\{c_1,c_2,c_3\}$, where $c_1 = (x_1 \vee \bar{x_2} \vee x_3)$, $c_2 = (\bar{x_1} \vee x_2 \vee \bar{x_3})$, and $c_3 = (\bar{x_1} \vee \bar{x_2} \vee x_3)$.

Lemma 2: The 3SAT instance $\mathcal{S}$ has a satisfying truth assignment if and only if the constructed REP instance $\mathcal{R}$ has an escaping solution with $d_{max} \leq 3$.

Proof: (if part) If the constructed REP instance $\mathcal{R}$ has an escaping solution with $d_{max} \leq 3$, we can obtain a satisfying truth assignment for the 3SAT instance $\mathcal{S}$. For each variable x_i, either the variable rectangle x_i escapes to the "True" boundary (the variable rectangle $\bar{x}_i$ escapes to the "False" boundary), or the variable rectangle x_i escapes to the "False" boundary (the variable rectangle $\bar{x}_i$ escapes to the "True" boundary), since if they escape to the same boundary, d_{max} is at least 4, due to the blockage rectangles. We set x_i to be true if the variable rectangle x_i escapes to the "True" boundary, and set it to be false otherwise. We claim this is a satisfying truth assignment of $\mathcal{S}$. For the sake of contradiction, suppose there is a clause c_i which is not satisfied by this assignment. Without loss of generality, let us take clause c_1 in Fig.4 as an example. Suppose the variables x_1, x_2 and x_3 are set to be false, true, and false, respectively, so that clause c_1 is not satisfied. In this case, the variable rectangles $\bar{x}_1$, x_2 and $\bar{x}_3$ escape to the "True" boundary, then it is easy to see that d_{max} will be 4 no matter how the three copies of clause rectangles for c_1 escape, which contradicts to the existence of an escaping solution with $d_{max} \leq 3$.

(only if part) If the 3SAT instance $\mathcal{S}$ has a satisfying truth assignment, we can obtain an escaping solution with $d_{max} \leq 3$ for the constructed REP instance. For each variable x_i, we let the two corresponding variable rectangles x_i and $\bar{x}_i$ escape to the "True" ("False") boundary and the "False" ("True") boundary, respectively, if variable x_i is set to be true (false). For each copy of clause rectangle c_i, we let it escape to the "True" boundary if this does not make d_{max} exceed 3, otherwise we let it escape to the "False" boundary. Now we have an escape solution for $\mathcal{R}$, and it is easy to see $d_{max} \leq 3$ for this escape solution. ■

Combining Lemma 1 and Lemma 2, we have proved that REP is NP-Complete.

Theorem 1: REP is NP-Complete.

IV. The Algorithm

In this section, we first show that the REP can be formulated into an Integer Linear Programming (ILP); then we demonstrate that a 4-approximation algorithm can be obtained by using Linear Programming (LP) relaxation and rounding technique. Furthermore, we introduce a more general version of REP with weights, namely, weighted REP, and show that the 4-approximation algorithm also works for weighted REP. Finally, an iterative refinement procedure is proposed as a postprocessing step to further improve the results.

A. ILP Formulation

We first introduce some notations in order to facilitate the ILP formulation. Given an REP instance $\mathcal{R}$:

- We introduce four 0-1 variables for each rectangle $r_i \in S$, namely, x_{il},x_{ir},x_{it} and x_{ib}, and we call them direction variables. The variable $x_{il}(x_{ir},x_{it},x_{ib})$ set to be 1 indicates that rectangle r_i escapes to the Left (Right, Top, Bottom) boundary of the rectangular region R.
- We extend the four boundary intervals of each rectangle $r_i \in S$ to the boundaries of the whole rectangular region R, so that a set of $O(n)$ horizontal and vertical cut-lines is obtained. Consequently, the rectangular region R is partitioned into a set P of $O(n^2)$ tiles by these cut-lines. Fig.5 shows an example consisting of four rectangles.
- For each rectangle $r_i \in S$, when it escapes to the boundary of R, it occupies a larger rectangular region, which is composed of the original region of r_i and the extension region after escaping. Let $r_{il}(r_{ir},r_{it},r_{ib})$ denote the rectangular region that r_i occupies after escaping to the Left (Right, Top, Bottom) boundary of R. Fig.5 illustrates the region r_{4r} and the region r_{4b} that rectangle r_4 occupies after escaping to the right boundary and the bottom boundary, respectively.

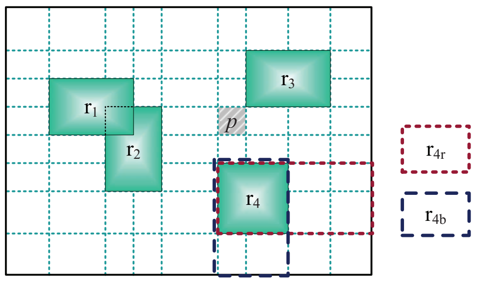


Fig. 5. The whole rectangular region R is partitioned into $O(n^2)$ small rectangular regions by the cut-lines obtained by extending the boundary intervals of the n rectangles.

Now we can add our constraints:

- **Direction Constraints**
  Each rectangle $r_i \in S$ can only choose one direction to escape, so we have the following set of constraints:
  $$x_{il} + x_{ir} + x_{it} + x_{ib} = 1, \quad \forall i = 1,2,\ldots,n.$$
- **Density Constraints**
  The density within each tile $p \in P$ should not exceed d_{max}, so we have the following set of constraints:
  $$\sum_{i,*:\, r_{i*} \text{ occupies } p} x_{i*} \leq d_{max}, \quad \forall p \in P,$$
  where '$*$' should be replaced by 'l' (Left), 'r' (Right), 't'(Top), or 'b' (Bottom), whichever is appropriate.

Let us take the tile p in Fig.5 as an example. It is easy to see that r_{1r}, r_{2r} and r_{4t} occupy tile p, so the density constraint for tile p is added as:

$$x_{1r} + x_{2r} + x_{4t} \leq d_{max}.$$

The objective is to minimize the density d_{max}, so now the ILP for the REP can be formulated as follows, with $O(n)$ variables and $O(n^2)$ constraints:

$$\begin{array}{lll} \text{Minimize} & d_{max} & \\ \text{Subject to} & & \\ & x_{il} + x_{ir} + x_{it} + x_{ib} = 1, & \forall i = 1,2,\ldots,n. \\ & \sum\limits_{i,*:\, r_{i*} \text{ occupies } p} x_{i*} \leq d_{max}, & \forall p \in P. \\ & x_{il}, x_{ir}, x_{it}, x_{ib} \in \{0,1\}, & \forall i = 1,2,\ldots,n. \end{array}$$

The optimal solution can be obtained by solving the above ILP.

B. A 4-Approximation Algorithm

Solving ILP is also an NP-Complete problem [2], and it can be very time-consuming when the problem is large. In the following, we show that a 4-approximation algorithm can be obtained by using LP relaxation and a special rounding technique.

We relax the ILP in Section IV-A into an LP as shown below, which can be efficiently solved by existing LP solvers.

$$\begin{array}{lll} \text{Minimize} & d_{max} & \\ \text{Subject to} & & \\ & x_{il} + x_{ir} + x_{it} + x_{ib} = 1, & \forall i = 1,2,\ldots,n. \\ & \sum\limits_{i,*:\, r_{i*} \text{ occupies } p} x_{i*} \leq d_{max}, & \forall p \in P. \\ & 0 \leq x_{il}, x_{ir}, x_{it}, x_{ib} \leq 1, & \forall i = 1,2,\ldots,n. \end{array}$$

We solve the LP by an LP solver, after which we get a fractional solution (if we are not lucky enough). For each rectangle $r_i \in S$, if x_{il} (x_{ir}, x_{it}, x_{ib}) has the largest value among the four direction variables of r_i, we say r_i's dominating escape direction is Left (Right, Top, Bottom), and we call x_{il} (x_{ir}, x_{it}, x_{ib}) the dominating variable of r_i. For example, if the LP solver produces a solution where $x_{il} = 0.1$, $x_{ir} = 0.2$, $x_{it} = 0.5$ and $x_{ib} = 0.2$ for rectangle r_i, the dominating escape direction of r_i is Top, and the dominating variable is x_{it}. We arbitrarily break the tie when two or more direction variables for r_i have the largest value. Our rounding technique works by, $\forall r_i \in S$, simply assigning 1 to the dominating variable, and assigning 0 to the other direction variables, i.e., each rectangle r_i's escape direction is set to be its dominating escape direction. This rounding technique gives us a 4-approximation algorithm, LPAPX, as described below.

```
ALGORITHM LPAPX(REP instance $\mathcal{R}$):
  1. Formulate $\mathcal{R}$ into an ILP.
  2. Relax the ILP into an LP.
  3. Solve the LP.
  4. $\forall r_i \in S$, assign $r_i$'s escape direction to be its
     dominating escape direction.
  5. Compute the resultant maximum density $d_{max}$.
```

Theorem 2: Given an REP instance $\mathcal{R}$, where each rectangle has 4 candidate choices of escape directions, LPAPX is a 4-approximation algorithm for $\mathcal{R}$.

Proof: Let $LP(d_{max})$ and $APX(d_{max})$ denote the maximum density d_{max} obtained by the LP (before rounding) and the algorithm LPAPX, respectively, and let $OPT(d_{max})$ denote the optimal maximum density. Obviously, $LP(d_{max}) \leq OPT(d_{max})$, since the solution space increases after LP relaxation.

Now we show that $APX(d_{max}) \leq 4LP(d_{max})$.

In step 3 of the algorithm LPAPX, we solve the LP and obtain a fractional solution, $\{(x_{il}, x_{ir}, x_{it}, x_{ib}), \forall i = 1,2,\ldots,n\}$. According to the density constraints of the LP,

$$\sum_{i,*:\, r_{i*} \text{ occupies } p} x_{i*} \leq LP(d_{max}), \quad \forall p \in P. \quad \langle 1 \rangle$$

Let $\{(\hat{x_{il}}, \hat{x_{ir}}, \hat{x_{it}}, \hat{x_{ib}}), \forall i = 1,2,\ldots,n\}$ denote the integral solution obtained after rounding, so we have

$$\sum_{i,*:\, r_{i*} \text{ occupies } p} \hat{x_{i*}} \leq APX(d_{max}), \quad \forall p \in P. \quad \langle 2 \rangle$$

There must be a tile $\hat{p} \in P$ such that the following equality holds:

$$\sum_{i,*:\, r_{i*} \text{ occupies } \hat{p}} \hat{x_{i*}} = APX(d_{max}). \quad \langle 3 \rangle$$

Let D denote the set of all the dominating variables. Now let us consider each term $\hat{x_{i*}}$ on the left-hand side of equation $\langle 3 \rangle$. There are two cases:
1). x_{i*} is a dominating variable ($x_{i*} \in D$). In this case, $x_{i*} \geq 0.25$. This is because $x_{il} + x_{ir} + x_{it} + x_{ib} = 1$, x_{i*} must be at least 0.25 to be the dominating variable. Thus, $\hat{x_{i*}} = 1 \leq 4x_{i*}$.
2). x_{i*} is not a dominating variable ($x_{i*} \notin D$). In this case, $\hat{x_{i*}} = 0$.

Thus, we rewrite equation $\langle 3 \rangle$ as follows:

$$\begin{aligned} APX(d_{max}) &= \sum_{i,*:\, r_{i*} \text{ occupies } \hat{p}} \hat{x_{i*}} \\ &= \sum_{i,*:\, r_{i*} \text{ occupies } \hat{p},\, x_{i*} \in D} \hat{x_{i*}} + \sum_{i,*:\, r_{i*} \text{ occupies } \hat{p},\, x_{i*} \notin D} \hat{x_{i*}} \\ &= \sum_{i,*:\, r_{i*} \text{ occupies } \hat{p},\, x_{i*} \in D} \hat{x_{i*}} + 0 \\ &\leq \sum_{i,*:\, r_{i*} \text{ occupies } \hat{p},\, x_{i*} \in D} 4x_{i*} \\ &\leq 4 \times \sum_{i,*:\, r_{i*} \text{ occupies } \hat{p}} x_{i*} \\ &\leq 4LP(d_{max}) \\ &\leq 4OPT(d_{max}) \end{aligned}$$

Therefore, LPAPX is a 4-approximation algorithm. ■

Based on the analysis in the proof of Theorem 2, it is easy to see that the approximation ratio of the algorithm LPAPX depends on the number of choices of the escape directions. For example, if each rectangle is only allowed to escape in two directions, LPAPX is a 2-approximation algorithm for REP. Therefore, we have the following immediate corollary:

Corollary 1: Given an REP instance $\mathcal{R}$, where each rectangle has α candidate choices of escape direction, LPAPX is an α-approximation algorithm for $\mathcal{R}$.

C. Weighted REP

In general, a rectangle does not necessarily contribute a unit density, and the density one rectangle contributes may vary when it escapes in different directions. Hence in this general version of REP, each rectangle $r_i \in S$ is associated with a weight vector $w_i = [w_{il}, w_{ir}, w_{it}, w_{ib}]$, where w_{il}, w_{ir}, w_{it} and w_{ib} denote the density rectangle r_i contributes when it escapes to the Left, Right, Top and Bottom, respectively. If two or more rectangles overlap with each other after escaping, the density of the overlapping region is calculated as the sum of the weights of the overlapping rectangles. We use weighted REP to denote this general version of REP.

Weighted REP
INSTANCE: A rectangular region R and a set S of n rectangles $\{r_1, r_2, \ldots, r_n\}$ residing within R. Each rectangle r_i is associated with a weight vector $w_i = [w_{il}, w_{ir}, w_{it}, w_{ib}]$, for $1 \leq i \leq n$.

QUESTION: Each rectangle $r_i \in S$ chooses a direction to escape, such that d_{max} is minimized.

Similarly, a weighted REP instance $\mathcal{R}$ can be formulated into the following ILP:

$$\begin{aligned}
&\text{Minimize} \quad d_{max} \\
&\text{Subject to} \\
&\qquad x_{il} + x_{ir} + x_{it} + x_{ib} = 1, && \forall i = 1, 2, \ldots, n. \\
&\qquad \sum_{i,*:\, r_{i*} \text{ occupies } p} w_{i*} \times x_{i*} \leq d_{max}, && \forall p \in P. \\
&\qquad x_{il}, x_{ir}, x_{it}, x_{ib} \in \{0, 1\}, && \forall i = 1, 2, \ldots, n.
\end{aligned}$$

This ILP can be relaxed into the following LP:

$$\begin{aligned}
&\text{Minimize} \quad d_{max} \\
&\text{Subject to} \\
&\qquad x_{il} + x_{ir} + x_{it} + x_{ib} = 1, && \forall i = 1, 2, \ldots, n. \\
&\qquad \sum_{i,*:\, r_{i*} \text{ occupies } p} w_{i*} \times x_{i*} \leq d_{max}, && \forall p \in P. \\
&\qquad 0 \leq x_{il}, x_{ir}, x_{it}, x_{ib} \leq 1, && \forall i = 1, 2, \ldots, n.
\end{aligned}$$

It is not difficult to figure out that the analysis in the proof of Theorem 2 still holds when we apply the algorithm LPAPX to a weighted REP instance. Therefore, we have the following theorem for weighted REP, the detailed proof of which is omitted.

Theorem 3: Given a weighted REP instance $\mathcal{R}$, where each rectangle has α candidate choices of escape directions, LPAPX is an α-approximation algorithm for $\mathcal{R}$.

In the PCB bus escape routing problem, it is possible that a bus occupies different number of layers for different escape directions. Take the bus in Fig.6 for example: One layer is enough to accommodate all the nets if the bus escapes to the left boundary or the right boundary, but two layers are needed if the bus escapes to the top boundary or the bottom boundary. This is due to the fact that the top boundary of the pin cluster's bounding box is much narrower than its right boundary. Weighted REP can exactly model this characteristic by the rectangle's weight vector. The weight vector of the rectangle for the bus in Fig.6 is $[1, 1, 2, 2]$.

D. Iterative Refinement

In this subsection, we present a greedy iterative refinement procedure as a postprocessing step to further improve the results obtained by the approximation algorithm LPAPX.

This greedy refinement procedure is performed iteratively. In each iteration, we try to re-escape all the rectangles one by one. When the re-escape for rectangle r_i is attempted, the escape directions of all the other rectangles are fixed. We try all the choices of escape directions for r_i and select the best one to be its new escape direction. The selection is done according to two criteria:

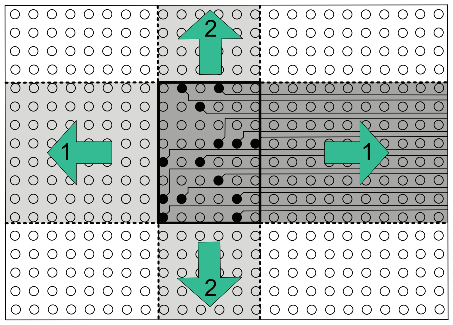


Fig. 6. Bus escape routes in different directions may occupy different numbers of layers.

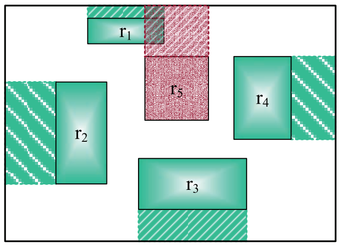


Fig. 7. The rectangle r_5 is set to escape to the top boundary since the least area is occupied in this way.

1) We pick the escape direction for r_i which results in smallest d_{max}.
2) If two or more escape directions for r_i result in the same d_{max}, the area occupied by the rectangle after escaping is used to break the tie. We pick the direction that results in less occupied area after escaping, potentially leaving more room for other rectangles, which is probably beneficial for the refinement of other rectangles.

Fig.7 shows an example with 5 rectangles. Suppose we are re-escaping rectangle r_5, with the escape directions of the other rectangles fixed. It is easy to see that d_{max} will be 2 no matter how r_5 escapes, but we decide to let it escape to the top boundary since the resultant rectangular region r_{5t} occupies the least area.

The pseudocode of the proposed iterative refinement procedure, GREEDYREFINE, is listed below.

```
ALGORITHM GREEDYREFINE(REP instance $\mathcal{R}$):
  terminate ← false;
  while ! terminate do
      for each rectangle $r_i \in S$ do
          Try all $r_i$'s escape directions;
          Pick the best one;
      if there is no improvement in this iteration then
          terminate ← true;
  return
```

E. Summary of the Algorithm

Our algorithm can be summarized as follows: Given a (weighted) REP instance $\mathcal{R}$, we first formulate it into an ILP, then apply LP relaxation and rounding technique to obtain an approximation solution, after which a greedy iterative refinement procedure is performed to improve the solution. The pseudocode of our algorithm flow, named REPAPX, is listed below.

TABLE I
REPAPX VS. SA ON INDUSTRIAL BUS ESCAPE TEST CASES

Test Cases	# Bus	REPAPX				
		LP (d_{max})	*APX* (d_{max})	*REP* (d_{max})	OPT	Runtime (sec)
Ex1	16	1	1	1	✓	0.012
Ex2	20	2	2	2	✓	0.032
Ex3	24	2	2	2	✓	0.016
Ex4	43	3.2	4	4	✓	0.060
Ex5	44	3.08	4	4	✓	0.064
Ex6	69	3	3	3	✓	0.440
Ex7	106	3.14	4	4	✓	1.328
Ex8	129	4.12	5	5	✓	2.224
Ex9	148	4.4	6	5	✓	2.640
Ex10	148	4.5	5	5	✓	2.932
# OPT		10/10				

```
ALGORITHM REPAPX((Weighted) REP instance $\mathcal{R}$):
    LPAPX($\mathcal{R}$);
    GREEDYREFINE($\mathcal{R}$);
    return
```

V. EXPERIMENTAL RESULTS

We implemented our approximation algorithm REPAPX in C++, with *Gurobi Optimizer* [1] employed as our LP solver. We tested REPAPX on 10 industrial PCB bus escape routing problems. The experiments are performed on a Linux workstation with two 3.0Ghz Intel Xeon CPUs and 4GB memory.

The experimental results are displayed in Table I. There are 10 test cases, which are derived from industrial PCB bus escape routing data. The column "*# Bus*" indicates the number of Buses (the number of rectangles in the corresponding REP instance). The results obtained by our REPAPX algorithm are listed in the multi-column "REPAPX". The "$LP(d_{max})$" column shows the d_{max} obtained after solving the LP (note that it might be a fractional number). The "$APX(d_{max})$" column shows the resultant d_{max} after rounding (LPAPX), while the "$REP(d_{max})$" column is the final maximum density d_{max} obtained after applying the iterative refinement procedure (GREEDYREFINE). In column "OPT", ✓means the solution is optimal and a $\times$ means it is not. Note that if $REP(d_{max}) = \lceil LP(d_{max}) \rceil$, we know the solution is optimal, since $OPT(d_{max}) \geq \lceil LP(d_{max}) \rceil$. For example, if $LP(d_{max}) = 4.5$, we know $d_{max} = 5$ is the best possible.

In each of the test cases, all the buses are allowed to escape in all the four directions. Although our algorithm REPAPX is a 4-approximation algorithm, the experimental results show that the performance of REPAPX is remarkably promising in practice, in the sense that each of the 10 test cases can be optimally solved within 3 seconds. Fig.8 shows the bus escape solution generated by REPAPX for test case *Ex2* with 20 rectangles, and the resultant maximum density $d_{max} = 2$.

VI. CONCLUDING REMARKS

In this paper, we introduce and study the Rectangle Escape Problem (REP), which originates in PCB bus escape routing. We prove that REP is NP-Complete, and propose a 4-approximation algorithm by using Linear Programming relaxation and rounding technique. This algorithm is also shown to work for weighed REP. Our algorithm is implemented and tested on a set of industrial PCB bus escape routing cases, and the results show that an optimal solution can be obtained within 3 seconds for each test case, which confirms the efficiency and effectiveness of our proposed algorithm.

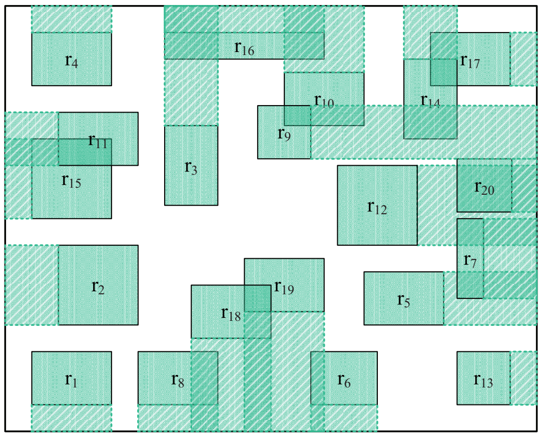


Fig. 8. The bus escape solution generated by REPAPX for test case Ex2 with 20 rectangles ($d_{max} = 2$).

REFERENCES

[1] Gurobi optimizer. http://www.gurobi.com.

[2] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1990.

[3] H. Harrer, H. Pross, T.-M. Winkel, W. D. Becker, H. I. Stoller, M. Yamamoto, S. Abe, B. J. Chamberlin, and G. A. Katopis. First- and second-level packaging for the IBM eserver z900. *IBM Journal of Research and Development*, 46:397–420, 2002.

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

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

[6] H. Kong, T. Yan, M. D. F. Wong, and M. M. Ozdal. Optimal bus sequencing for escape routing in dense PCBs. In *ICCAD '07: Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design*, pages 390–395, 2007.

[7] D. T. Lee. Maximum clique problem of rectangle graphs. *Advances in Computing Research*, 1:91–107, 1983.

[8] M. M. Ozdal. *Routing algorithms for high-performance VLSI packaging*. PhD thesis, University of Illinois at Urbana-Champaign, 2005.

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

[10] T.-M. Winkel, W. D. Becker, H. Harrer, H. Pross, D. Kaller, B. Garben, B. J. Chamberlin, and S. A. Kuppinger. First- and second-level packaging for the z990 processor cage. *IBM Journal of Research and Development*, 48:379–394, 2004.