NP-Completeness and an Approximation Algorithm for Rectangle Escape Problem With Application to PCB Routing

Qiang Ma and Martin D. F. Wong, *Fellow, IEEE*

***Abstract*—In this paper, we introduce and study the rectangle escape problem (REP), which is motivated by printed circuit board (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 programming (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. In addition, an iterative refinement procedure is proposed as a postprocessing step to further improve the results. Our approximation algorithm is also shown to work for more general versions of REP: weighted REP and simultaneous REP. 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 several seconds for each of the test cases.**

***Index Terms*—Approximation algorithm, linear programming relaxation, NP-completeness, printed circuit board (PCB) routing.**

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 projecting the rectangle onto one of the four boundaries of R, namely, left, right, top, or bottom. The resultant rectangle (including the region of the original rectangle) is called a projection rectangle. 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 projection rectangles containing this point, and the point with largest density defines the maximum density over R. Fig. 1 illustrates the REP; Fig. 1(a) shows the input rectangles in S, while Fig. 1(b) and (c) gives 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 2 and 3, respectively. 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 printed circuit boards (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], [10]–[12]. 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], [5], [7]. 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. A projection rectangle of a bus is defined as the rectangular region including the bounding box of the pin cluster of the bus as well as the extension part obtained by projecting the bounding box to one of the boundaries. The shaded region in Fig. 2 demonstrates a projection rectangle of a bus. Fig. 3 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. In [6], Kong *et al.* compared their bus projection-based escape routing solution with the net-based escape routing solution and observed that the bus projection-based solution usually resulted in shorter wire length and occupied smaller routing space. Moreover, the wire length of the nets within a bus are usually required to be approximately identical [13], due to the timing issue. The bus projection-based routing style produces much more balanced wire length, making length-matching routing an easier problem [13].

When the escape routes of two buses conflict, they have to be routed on different layers. Fig. 4 demonstrates two types

Manuscript received September 12, 2011; revised December 21, 2011 and February 28, 2012; accepted March 19, 2012. Date of current version August 22, 2012. This work was supported in part by the National Science Foundation, under Grant CCF-1017516. The preliminary version of this article appeared in the Proceedings of ASPDAC'11 [9]. This paper was recommended by Associate Editor E. Young.

Q. Ma is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA, and also with Synopsys, Inc., Mountain View, CA 94043 USA (e-mail: qma@synopsys.com).

M. D. F. Wong is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: mdfwong@illinois.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2012.2193581

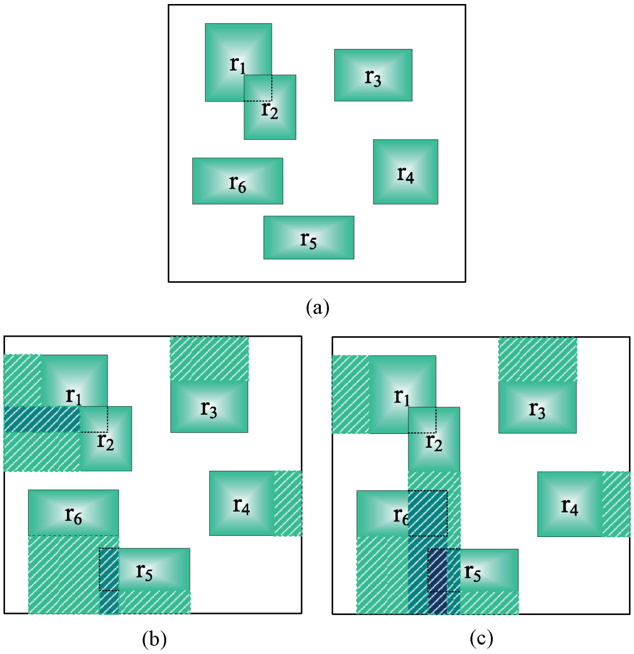


Fig. 1. Illustration of REP. (a) Input rectangles. (b) Escape solution with maximum density equal to 2. (c) Escape solution with maximum density equal to 3.

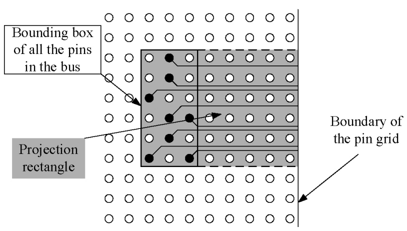


Fig. 2. Projection rectangle is obtained by projecting the bounding box of the pin cluster of the bus to one of the boundaries of the component.

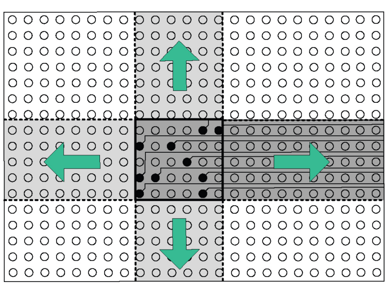


Fig. 3. Four projection rectangles of a bus and its escape routing to the right boundary of the component.

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 theoretically it is only a lower bound of the necessary number of layers). Therefore, in our REP, we try to minimize the maximum density. Note that a whole PCB board contains a number of components (pin grids), and by solving the REP for each component, we can obtain a bus escape planning for the whole PCB board. After that, the corresponding buses of different components

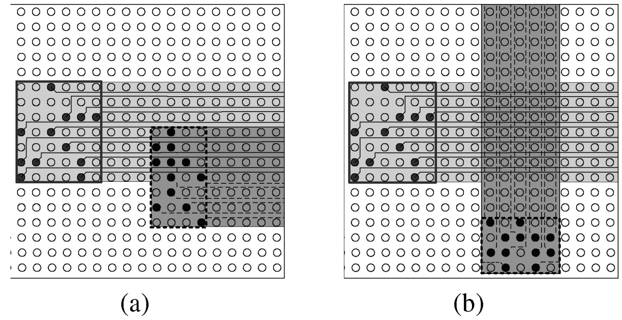


Fig. 4. Routes of two buses conflict. (a) One type of conflict. (b) Another type of conflict.

can further be connected and the actually layer assignment can be performed.

Our contributions in this paper can be summarized as follows.

1) We prove that the REP is NP-complete.
2) A 4-approximation algorithm for REP is developed.
3) This approximation algorithm is also shown to work for more general versions of REP: weighted REP and simultaneous REP.
4) 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 several 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. The NP-completeness of the REP is proved in Section III, and our approximation algorithm for the REP is presented in Section IV. Experimental results are reported in Section VI 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, which is defined as follows.

Definition 1: Maximum density $d_{\max}$: In the rectangular region R, the density at a point is the number of projection rectangles containing this point, and the point with the largest density defines the maximum density $d_{\max}$ over R.

We can now define the REP 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 $d_{\max} \le k$?

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 [8] 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 $1 \le i \le m$.

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 $d_{\max} \le 3$. 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. 5).
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. 5 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. 5 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 $i \neq j$.

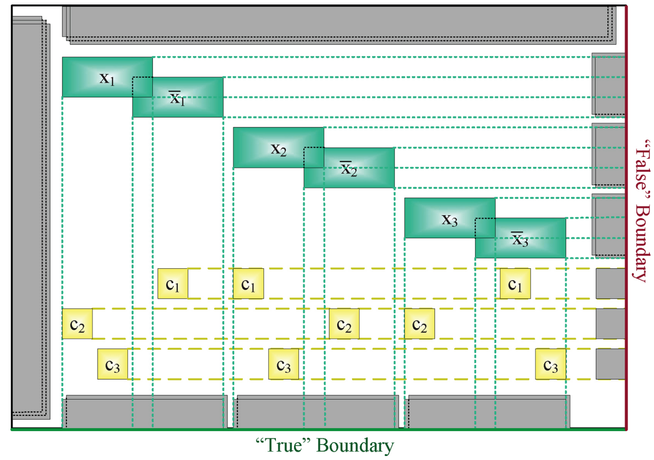


Fig. 5. REP instance $\mathcal{R}$ constructed from the 3SAT instance $\mathcal{S}$ ($\mathcal{S}$ contains three variables $\{x_1, x_2, x_3\}$ and three clauses $\{c_1, c_2, c_3\}$, where $c_1 = (x_1 \vee \bar{x_2} \vee x_3)$, $c_2 = (\bar{x_1} \vee x_2 \vee \bar{x_3})$, and $c_3 = (\bar{x_1} \vee \bar{x_2} \vee x_3)$).

It is easy to see that the reduction can be done in polynomial time with respect to the size of the 3SAT instance $\mathcal{S}$, so we have the following Lemma.

Lemma 2: The reduction from a general 3SAT instance $\mathcal{S}$ to a REP instance $\mathcal{R}$ can be done in polynomial time with respect to the size of the 3SAT instance $\mathcal{S}$.

Fig. 5 shows the REP instance $\mathcal{R}$ constructed from a concrete 3SAT instance $\mathcal{S}$ with three variables $\{x_1, x_2, x_3\}$ and three 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 3: 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 $d_{\max} \le 3$.

Proof: (If part): If the constructed REP instance $\mathcal{R}$ has an escaping solution with $d_{\max} \le 3$, 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 that this is a satisfying truth assignment of $\mathcal{S}$. For the sake of contradiction, suppose there is a clause c_i that is not satisfied by this assignment. Without loss of generality, let us assume that clause c_i contains variable x_j. According to our construction of the REP instance, one of the three copies of clause rectangle c_i is placed below the variable rectangle $\bar{x}_j$. Since clause c_i is not satisfied, x_j must be set to false, which indicates that the variable rectangle x_j escapes to the False boundary and the variable rectangle $\bar{x}_j$ escapes to the True boundary. Therefore, this copy of clause rectangle c_i can only escape to the False boundary, as escaping to the True boundary makes the local density become 4 (two blockages rectangles along the True boundary, the variable rectangle $\bar{x}_j$, and this copy of clause rectangle c_i). Based on the same argument, the other two copies of clause rectangle c_i have to escape to the False boundary, too. However, if all the three copies of clause rectangle c_i escape to the False boundary, it also makes the local density become 4 (one blockage rectangle along the False boundary and the three copies of clause rectangle c_i). Now, we can conclude that $d_{\max}$ will be at least 4 no matter how we escape the three copies of clause rectangles c_1, 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 the three Lemmas, we have proved that REP is NP-complete.

Theorem 1: REP is NP-complete.

IV. 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 $\mathcal{R}$, the following results.

1) 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.
2) 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. 6 shows an example consisting of four rectangles.

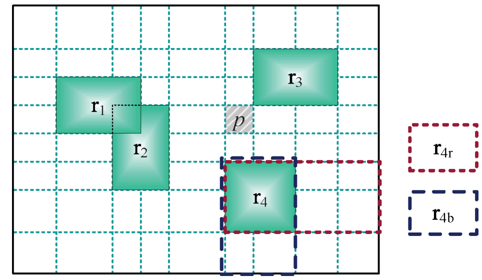


Fig. 6. Whole rectangular region R is partitioned into $O(n^2)$ small rectangular regions by the cut-lines obtained by extending the boundary intervals of the n rectangles.

3) 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. 6 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.

Now we can add our constraints as follows.

1) *Direction constraints*: Each rectangle $r_i \in S$ can only choose one direction to escape, so we have the following set of constraints:

$$x_{il} + x_{ir} + x_{it} + x_{ib} = 1, \quad \forall i = 1, 2, \ldots, n.$$

2) *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. 6 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 with $O(n)$ variables and $O(n^2)$ constraints as follows:

Minimize $d_{\max}$
Subject to

$$x_{il} + x_{ir} + x_{it} + x_{ib} = 1, \quad \forall i = 1, 2, \ldots, n.$$

$$\sum_{i,*:\, r_{i*} \text{ occupies p}} x_{i*} \leq d_{\max}, \quad \forall p \in P.$$

$$x_{il}, x_{ir}, x_{it}, x_{ib} \in \{0, 1\}, \quad \forall i = 1, 2, \ldots, n.$$

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

Algorithm 1 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_is escape direction to be its dominating escape direction.
5. Compute the resultant maximum density $d_{\max}$.

B. 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

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

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_is 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_is escape direction is set to be its dominating escape direction. This rounding technique gives us a 4-approximation algorithm, LpApx (Algorithm 1).

Theorem 2: Given an REP instance $\mathcal{R}$, where each rectangle has four 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 Algorithm 1, 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

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

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

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

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

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

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 (3). 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 (3) as follows:

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

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. Tightness of the Approximation Ratio

In the following, we show that the approximation ratio 4 is tight by constructing a REP instance, where the density $d_{\max}$ of the solution obtained by LpApx is exactly four times that of the optimal solution. Fig. 7 shows a REP instance containing ten rectangles. It is easy to see that the density $d_{\max}$ of the optimal solution of this instance is 2, which is determined by the overlapping of r_9 and r_{10}. We show in the following that we can obtain a solution with $d_{\max} = 8$ using LpApx. We use an

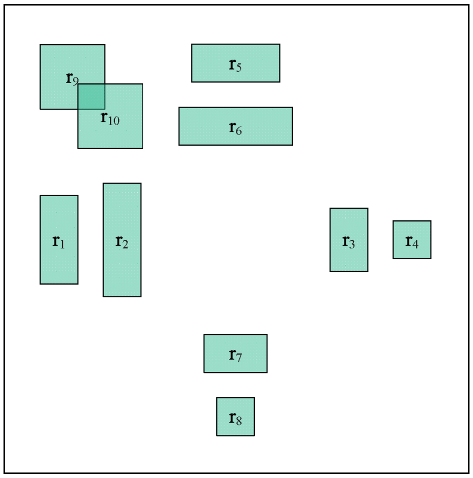


Fig. 7. REP instance containing ten rectangles. The density $d_{\max}$ of its optimal solution is 2.

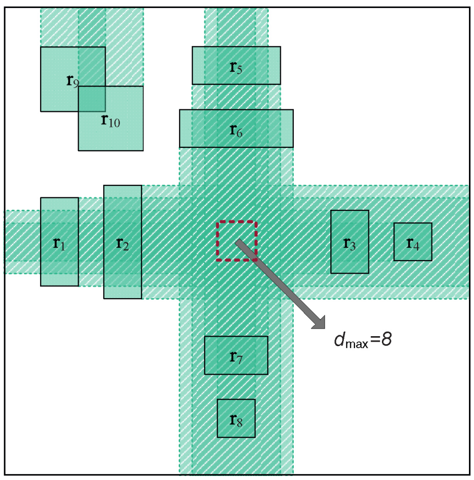


Fig. 8. 4-approximation solution of the REP instance in Fig. 7.

LP solver to solve the relaxed ILP of this REP instance. Note that the set of optimal solutions of this relaxed ILP includes a fractional solution, where $x_{il} = x_{ir} = x_{it} = x_{ib} = 0.25$, $\forall i = 1, 2, \ldots, 8$, $r_{it} = 1$, $\forall i = 9, 10$. This corresponds to an escape solution, where r_i escapes a "quarter" to each of the four directions, $\forall i = 1, 2, \ldots, 8$, r_9 and r_{10} escape to the top. It is possible that the LP solver returns the above-mentioned fractional solution. LpApx then decides an escape direction for each rectangle using the rounding technique. As there is no dominating escape direction for $r_1, r_2, \ldots, r_8$, LpApx will arbitrarily pick an escape direction for them. In the worst case, the escape directions of r_1 and r_2 (r_3 and r_4, r_5 and r_6, r_7 and r_8) are set to be right (left, bottom, top), which makes the density at the center as 8 (four times of that of the optimal solution), as illustrated in Fig. 8. This example shows the tightness of the approximation ratio of our Algorithm 1.

D. Iterative Refinement

In this section, we present a greedy iterative refinement procedure as a postprocessing step to further improve the results obtained by the approximation Algorithm 1.

Algorithm 2 GreedyRefine (REP instance $\mathcal{R}$)

terminate $\leftarrow$ *false*;
while ! *terminate* **do**
  for each rectangle $r_i \in S$ **do**
    Try all r_is escape directions;
    Pick the best one;
  if there is no improvement in this iteration **then**
    terminate $\leftarrow$ *true*;
return

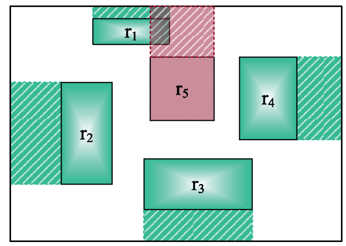


Fig. 9. Rectangle r_5 is set to escape to the top boundary since the least area is occupied in this way.

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:

1) we pick the escape direction for r_i that 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. 9 shows an example with five 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 in Algorithm 2.

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 in Algorithm 3.

Algorithm 3 RepApx [(Weighted) REP instance $\mathcal{R}$]

LpApx($\mathcal{R}$);
GreedyRefine($\mathcal{R}$);
return

V. Extensions

In this section, we show that our algorithm can be easily extended to handle more general versions of REP, namely, weighted REP and simultaneous REP.

A. 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:

Minimize $d_{\max}$
Subject to

$$x_{il} + x_{ir} + x_{it} + x_{ib} = 1, \qquad \forall i = 1, 2, \ldots, n.$$

$$\sum_{i,*\,:\,r_{i*} \text{ occupies p}} w_{i*} \times x_{i*} \leq d_{\max}, \forall p \in P.$$

$$x_{il}, x_{ir}, x_{it}, x_{ib} \in \{0, 1\}, \qquad \forall i = 1, 2, \ldots, n.$$

This ILP can be relaxed into the following LP:

Minimize $d_{\max}$
Subject to

$$x_{il} + x_{ir} + x_{it} + x_{ib} = 1, \qquad \forall i = 1, 2, \ldots, n.$$

$$\sum_{i,*\,:\,r_{i*} \text{ occupies p}} w_{i*} \times x_{i*} \leq d_{\max}, \forall p \in P.$$

$$0 \leq x_{il}, x_{ir}, x_{it}, x_{ib} \leq 1, \qquad \forall i = 1, 2, \ldots, n.$$

It is not difficult to figure out that the analysis in the proof of Theorem 2 still holds when we apply Algorithm 1 to a weighted REP instance. Thus, we have the following theorem.

Theorem 3: Given a weighted REP instance $\mathcal{R}$, where each rectangle has four candidate choices of escape directions, LpApx is an 4-approximation algorithm for $\mathcal{R}$.

Proof: Let us reuse the notations introduced in the proof of Theorem 2. We need to show that $\text{APX}(d_{\max}) \leq 4\text{LP}(d_{\max})$.

Similarly, after rounding, there must be a tile $\hat{p} \in P$ such that the following equality holds:

$$\sum_{i,*\,:\,r_{i*} \text{ occupies } \hat{p}} w_{i*} \times \hat{x_{i*}} = \text{APX}(d_{\max}). \tag{4}$$

Note that the following two statements hold too:

1) $\hat{x_{i*}} = 1 \leq 4x_{i*}$, if x_{i*} is a dominating variable ($x_{i*} \in D$);
2) $\hat{x_{i*}} = 0$, if x_{i*} is not a dominating variable ($x_{i*} \notin D$).

Thus, we rewrite (4) as follows:

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

Therefore, LpApx is a 4-approximation algorithm for a weighted REP instance. ■

Similarly, we have the following immediate corollary.

Corollary 2: 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. For example, take the bus in Fig. 10. 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. 10 is $[1, 1, 2, 2]$.

B. Simultaneous REP

Simultaneous REP models the simultaneous bus escape problem in PCB routing. It considers two components, each of which contains the same set of buses. The corresponding buses in the two components need to be connected. Fig. 11 shows an example of simultaneous REP. The rectangle (representing a bus) r_1 (r_2, r_3, r_4) in the left component needs to be connected with the rectangle r'_1 (r'_2, r'_3, r'_4) in the right component.

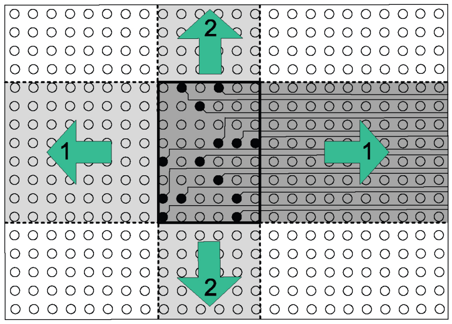


Fig. 10. Bus escape routes in different directions may occupy different numbers of layers.

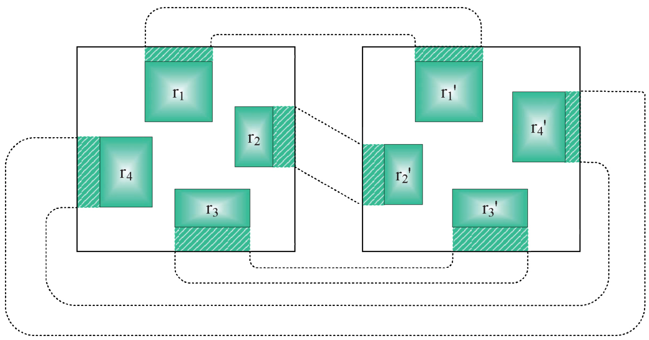


Fig. 11. Simultaneous REP.

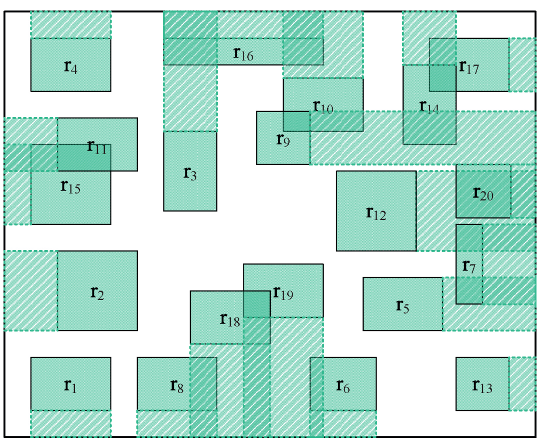


Fig. 12. Bus escape solution generated by Algorithm 3 for test case Ex2 with 20 rectangles ($d_{\max} = 2$).

This imposes a set of constraints on the escape directions of rectangles in the two components.

1) If rectangle r_i in the left component escapes to the right, the corresponding rectangle r_i' in the right component is preferred to escape to the left.
2) If rectangle r_i in the left component escapes to the left, the corresponding rectangle r_i' in the right component is preferred to escape to the right.
3) If rectangle r_i in the left component escapes to the top, the corresponding rectangle r_i' in the right component is preferred to escape to the top.
4) If rectangle r_i in the left component escapes to the bottom, the corresponding rectangle r_i' in the right component is preferred to escape to the bottom.

This is what we observed from industrial manual solutions of PCB bus escape routing, and satisfying these constraints usually make things easier and more smooth for the length-matching problem and the layer assignment problem in a later phase. Therefore, in simultaneous REP, we need to minimize the maximum density $d_{\max}$ over the two components, while satisfying the above constraints.

We show in the following that our LP relaxation and rounding technique can be easily extended and adapted to handle simultaneous REP.

Let $\{x_{il}, x_{ir}, x_{it}, x_{ib}\}$ be the direction variables of rectangle r_i in the left component, and let $\{x_{il}', x_{ir}', x_{it}', x_{ib}'\}$ be the direction variables of the corresponding rectangle r_i' in the right component, $\forall i = 1, 2, \ldots, n$. Simultaneous REP can be formulated into the following ILP:

Minimize $d_{\max}$

Subject to

$$
\begin{aligned}
& x_{il} + x_{ir} + x_{it} + x_{ib} = 1, && \forall i = 1, 2, \ldots, n \\
& x_{il}' + x_{ir}' + x_{it}' + x_{ib}' = 1, && \forall i = 1, 2, \ldots, n \\
& \sum_{i,*\,:\, r_{i*} \text{ occupies p}} x_{i*} \le d_{\max}, && \forall p \in P \\
& \sum_{i,*\,:\, r_{i*}' \text{ occupies p}'} x_{i*}' \le d_{\max}, && \forall p' \in P' \\
& x_{il} = x_{ir}', && \forall i = 1, 2, \ldots, n \\
& x_{ir} = x_{il}', && \forall i = 1, 2, \ldots, n \\
& x_{it} = x_{it}', && \forall i = 1, 2, \ldots, n \\
& x_{ib} = x_{ib}', && \forall i = 1, 2, \ldots, n \\
& x_{il}, x_{ir}, x_{it}, x_{ib} \in \{0, 1\}, && \forall i = 1, 2, \ldots, n \\
& x_{il}', x_{ir}', x_{it}', x_{ib}' \in \{0, 1\}, && \forall i = 1, 2, \ldots, n.
\end{aligned}
$$

Similarly, we can relax the above ILP into LP and use an LP solver to solve it. We can then apply the same rounding technique to obtain an approximation solution. It is easy to figure out that the approximation ratio still holds. Therefore, we have the following theorem, the detailed proof of which is omitted.

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

VI. Experimental Results

We implemented our approximation algorithm RepApx in C++, with *Gurobi Optimizer* [1] employed as our LP solver. To validate our proposed approach, we perform two sets of experiments on industrial PCB bus escape routing problems. The experiments are performed on a Linux workstation with two 3.0 GHz Intel Xeon CPUs and 4 GB memory.

In the first set of experiments, we compare the performance of RepApx with the ILP approach as well as a greedy approach on a set of unweighted REP test cases. In the ILP approach, the ILP is directly solved by the ILP solver (Gurobi Optimizer) without using LP relaxation. In the greedy approach, we directly apply the GreedyRefine procedure from the beginning.

TABLE I
COMPARISON WITH ILP AND GREEDY APPROACH

Test Cases	# Bus	d_{max}					Runtime (s)		
		RepApx			ILP	Greedy			
		LP	APX	REP			RepApx	ILP	Greedy
Ex1	16	1	1	1	1	1	0.012	0.063	0.008
Ex2	20	2	2	2	2	2	0.032	0.241	0.011
Ex3	24	2	2	2	2	2	0.016	0.078	0.015
Ex4	43	3.2	4	4	4	5	0.060	0.432	0.028
Ex5	44	3.08	4	4	4	5	0.064	0.671	0.032
Ex6	69	3	3	3	3	4	0.440	1.567	0.117
Ex7	106	3.14	4	4	4	5	1.328	51.79	0.329
Ex8	129	4.12	5	5	5	6	2.224	95.63	0.884
Ex9	148	4.4	6	5	5	6	2.640	118.6	1.263
Ex10	148	4.5	5	5	5	6	2.932	162.5	1.411
# OPT		10/10			10/10	3/10	–		

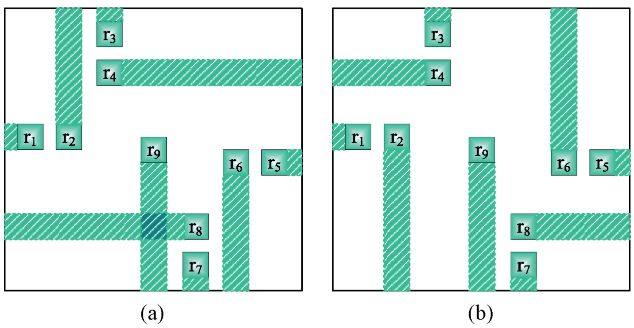


Fig. 13. Example REP instance with nine rectangles. (a) Solution obtained by the greedy approach. (b) Optimal solution.

TABLE II
EXPERIMENTAL RESULTS ON WEIGHTED TEST CASES

Test Cases	# Bus	RepApx				
		LP (d_{max})	APX (d_{max})	REP (d_{max})	OPT (d_{max})	Runtime (s)
Ex1w	16	2	2	2	2	0.016
Ex2w	20	3	3	3	3	0.042
Ex3w	24	3.3	4	4	4	0.435
Ex4w	43	4.2	5	5	5	0.117
Ex5w	44	4.55	6	5	5	0.120
Ex6w	69	5.12	7	6	6	1.262
Ex7w	106	6.34	7	7	7	3.781
Ex8w	129	7.16	9	8	8	5.526
Ex9w	148	7.32	9	8	8	5.903
Ex10w	148	7.45	8	8	8	6.650
# OPT		10/10				

The experimental results are displayed in Table I. There are ten unweighted REP 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 Algorithm 3 are listed in the multicolumn "RepApx." The "LP" column shows the d_{max} obtained after solving the LP (note that it might be a fractional number). The "APX" column shows the resultant d_{max} after rounding (LpApx), while the "REP" column is the final maximum density d_{max} obtained after applying the iterative refinement procedure (GreedyRefine). The results obtained by the ILP approach and the greedy approach are listed in the "ILP" column and the "Greedy" column, respectively. The ILP approach produces the optimal solution. We can see that our proposed approximation Algorithm 2 obtains the same d_{max} value with the ILP approach for all the test cases, i.e., it optimally solves all the test cases, while the greedy approach only solves three test cases optimally. As for the runtime comparison, we can see that our approximation Algorithm 3 is about 50× faster than directly solving the ILP for large test cases. Although the runtime of the ILP solver seems acceptable, solving ILP is NP-complete and there is no guarantee. When the problem size grows larger, the ILP solver can become unexpectedly slow. Fig. 12 shows the bus escape solution generated by Algorithm 3 for test case *Ex2* with 20 rectangles, and the resultant maximum density $d_{max} = 2$. As we can see from the table, the gap between the results obtained by the greedy method and that by the proposed approach is not large. However, the greedy method can get stuck in some cases, due to its greedy nature. The example in Fig. 13 gives an illustration. Fig. 13(a) shows the solution obtained by the greedy approach. Suppose the escape directions of the buses are determined in the order of $r_1, r_2, \ldots, r_9$ in the greedy approach. r_2 escapes to the top boundary and r_4 escapes to the right boundary. Then, r_6 and r_8 can only escape to the bottom boundary and left boundary, respectively, since they will overlap with other buses otherwise. We will then notice that r_9 will overlap with other buses no matter in which direction it escapes, so that we obtain an escape solution with $d_{max} = 2$. We then re-escape all the buses one by one, but it is easy to see that the greedy method gets stuck at this escape solution, as there is no improvement that can be obtained by changing the escape direction of the buses. Fig. 13(b) shows the optimal escape solution with $d_{max} = 1$. It is possible that the greedy method can achieve the optimal solution by using a "good" ordering in which the buses are escaped. However, in general, finding a good ordering by itself is a very difficult problem (we cannot afford to enumerate all the possible orderings). Moreover, it is also possible that such a good ordering does not exist at all, i.e., no ordering gives us the optimal solution if the buses are escaped one by one (this is the nature of the greedy algorithm).

In the second set of experiments, we test our approximation Algorithm 3 on a set of weighted REP test cases. For a bus, the weight associated with each escape direction is set to be the number of layers this bus occupies if it escapes to that direction. The results are shown in Table II. Similar with that in Table I, the columns "LP," "APX," and "REP" show the $d_{\max}$ obtained after solving the LP, after rounding and after iterative refinement, respectively. The column "OPT" shows the optimal $d_{\max}$ computed by ILP. We can see that Algorithm 3 solves each of the weighted REP test cases optimally within several seconds.

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 Algorithm 3 is remarkably promising in practice, in the sense that each of the test cases (both the unweighted and weighted) can be optimally solved within several seconds.

VII. Conclusion

In this paper, we introduced and studied the REP, which originates in PCB bus escape routing. We proved that REP is NP-complete, and proposed a 4-approximation algorithm by using LP relaxation and rounding technique. This algorithm is also shown to work for weighed REP and simultaneous REP. Our algorithm was implemented and tested on a set of industrial PCB bus escape routing cases, and the results showed that an optimal solution can be obtained within several seconds for each test case, which confirms the efficiency and effectiveness of our proposed algorithm.

References

[1] Gurobi. (2012, Feb. 25). *Gurobi Optimizer* [Online]. Available: http://www.gurobi.com

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 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 J. Res. Develop.*, vol. 46, nos. 4–5, pp. 397–420, Jul.–Sep. 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," in *Proc. 47th Annu. DAC*, 2010, pp. 212–217.

[5] H. Kong, T. Yan, and M. D. F. Wong, "Automatic bus planner for dense PCBs," in *Proc. 46th Annu. DAC*, 2009, pp. 326–331.

[6] H. Kong, T. Yan, and M. D. F. Wong, "Optimal simultaneous pin assignment and escape routing algorithm for rectangle escape problem with appilication to PCB routing," in *Proc. 15th ASPDAC*, 2010, pp. 275–280.

[7] H. Kong, T. Yan, M. D. F. Wong, and M. M. Ozdal, "Optimal bus sequencing for escape routing in dense PCBs," in *Proc. IEEE/ACM ICCAD*, Nov. 2007, pp. 390–395.

[8] D. T. Lee, "Maximum clique problem of rectangle graphs," in *Advances in Computing Research*, vol. 1, F. P. Preparata, Ed. JAI Press, 1983, pp. 91–107.

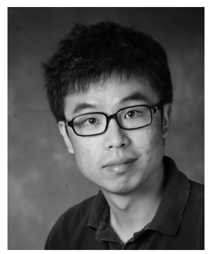
[9] Q. Ma, H. Kong, M. D. F. Wong, and E. F. Y. Young, "A provably good approximation algorithm for rectangle escape problem with application to PCB routing," in *Proc. 16th ASPDAC*, 2011, pp. 843–848.

[10] M. M. Ozdal, "Routing algorithms for high-performance VLSI packaging," Ph.D. dissertation, Dept. Comput. Sci., Univ. Illinois Urbana-Champaign, Urbana, 2005.

[11] 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. IEEE/ACM ICCAD*, Nov. 2005, pp. 759–766.

[12] 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 J. Res. Develop.*, vol. 48, no. 4, pp. 379–394, 2004.

[13] T. Yan and M. D. F. Wong, "BSG-Route: A length-constrainted routing scheme for general planar topology," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 28, no. 11, pp. 1679–1690, Nov. 2009.



Qiang Ma received the B.E. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 2006, the M.Phil. degree in computer science from the Chinese University of Hong Kong, Shatin, Hong Kong, in 2008, and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign, Urbana, in 2012.

He is currently a Senior Research and Development Engineer with Synopsys, Inc., Mountain View, CA. His current research interests include physical design of chips, packages, and printed circuit boards.

Martin D. F. Wong (F'06) received the B.S. degree from the University of Toronto, Toronto, ON, Canada, and the M.S. degree from the University of Illinois at Urbana-Champaign (UIUC), Urbana, both in mathematics, and the Ph.D. degree in computer science from UIUC in January 1987.

From 1987 to 2002, he was a Faculty Member with the Department of Computer Science, University of Texas at Austin, Austin. Since 2002, he has been with UIUC, where he is currently a Professor of electrical and computer engineering with the Department of Electrical and Computer Engineering. He has published approximately 400 technical papers and graduated 41 Ph.D. students in the area of electronic design automation (EDA).

Dr. Wong has won several Best Paper Awards for his works in EDA. He was a Technical Program Committee Member of leading EDA conferences. He was also an Editorial Board Member of the IEEE Transactions on Computers, the IEEE Transactions on Computer-Aided Design, and the *ACM Transactions on Design Automation of Electronic Systems*. He was an IEEE Distinguished Lecturer from 2005 to 2006.