On Channel Routing Problems with Interchangeable Terminals

SPYROS TRAGOUDAS
Department of Computer Science, Southern Illinois University, Carbondale, IL 62901

(Received July 29, 1992; Revised January 29, 1993)

The use of programmable logic cells in VLSI design allows the terminals on these cells to be interchanged since their geometries are programmable. Recently, many exact algorithms and heuristics have been proposed for channel routing with interchangeable terminals [18, 25, 4, 11, 12, 20, 17, 3]. Various optimization problems have also been shown to be NP-hard [25, 23]. In this paper, we consider channels with exits. Let \( m, D \) be the number of terminals in the channel and the maximum number of terminals on a net, respectively. We present an \( O(m) \) algorithm that obtains optimal density for channels with exits that have one cell on each side. The existing algorithm for this problem [5] guarantees only an approximate density. Moreover, if one of the two cells has fixed terminals, we show that the density minimization problem is NP-hard. The latter problem was introduced in [5]. For instances with any number of cells we present an \( O(m) \) time algorithm for the via minimization problem, an \( O(m D) \) algorithm for the problem of finding a maximum planar subset of nets in a channel, and an \( O(m) \) algorithm to determine whether the channel adopts external-internal layout. Also for the special case where there exists one cell per side, we present an alternative algorithm that finds a maximum planar set of nets in \( O(m) \) time.

Key Words: Algorithms; VLSI Layout; Channel routing; Pin assignment; Density; NP-completeness

1. INTRODUCTION

*Channel routing* is a fundamental problem in VLSI design. In a channel routing problem the input consists of a collection of cells placed on adjacent positions and partitioned in two groups called the top and the bottom channel side, respectively. A rectilinear channel is formed between the two channel sides. Each cell contains terminals (otherwise called pins). Terminals are labeled according to the net they belong. Terminals labeled with zero do not belong to any net and we call them zero terminals. We call the remaining terminals nonzero terminals.

Algorithms to route channels with fixed terminals have been proposed by several authors [19, 2, 6, 21, 27]. Recently, a new model for channel routing has been proposed [12] in which we are considering that some or all of the cells on the channel sides consist of terminals whose relative positions within the cell is not fixed, i.e., the terminals can be interchanged in their cells. This is allowed if the cells have been implemented with programmable logic since their geometries are programmable. Programmable logic cells (i.e., ROM's and PLA's) are widely used in VLSI design because of their structural regularity and flexibility [12]. Terminals can be interchanged only if they are in the same cell. A methodology for performing the routing in this type of channels consists of two steps: We first interchange terminals within each cell in order to minimize some routing requirements (e.g., density, number of layers, number of vias etc.) and second, route the “new” channel.

The channel routing problem with interchangeable terminals in their cells has been extensively studied. Kobyashi and Drozd [12] present heuristics for aligning terminals, breaking cycles in the *vertical constraints graph*, and merging nets. Lin and Sahni [18] present a linear time algorithm that obtains *maximum terminal alignment*. We say that two terminals are aligned if they are on the same net, on opposite channel sides and on the same vertical column of the underlying rectilinear grid. Hou and Chen [11] and Pedram, Marek-Sadowska and Kuh [20] and Leong and Liu [17] present heuristics that result in reduced
number of tracks required for routing. These heuristics attempt to minimize the channel width by minimizing the channel density. Sahni and Wu [23] showed that it is NP-complete [8] to determine whether there is an interchange of terminals that results in an acyclic vertical constraints graph (VCG) or whether a terminal interchange can reduce the length of any path in an acyclic VCG to be less than an integer k. In [25] Tollis and Tragoudas present a linear time algorithm to determine whether the channel is river routable, they showed that the problem of minimizing the density is NP-hard in general, but they presented polynomial time algorithms for the special case of channels with two terminal nets and channels with at most one terminal per cell for each net. Liao and Sarrafzadeh [16] present an algorithm that detects whether a switchbox adopts a river routable solution. Leong [15] presents polynomial time algorithms that minimize the density in channel instances that satisfy all the following constraints: (a) each net appears in exactly one cell on the top side and in exactly one cell on the bottom side, (b) the number of cells on the top side is equal to the number of cells on the bottom side and is equal to the number of partitions for the nets, and (c) there are no zero terminals.

Cong [4], and Cong and Khoo [5] examine the density minimization problem on a simpler terminal permutation problem where the channels are interchangeable anywhere on the channel side they reside. Therefore they consider channels that consist of only one cell per channel side. The algorithms in [4, 5] handle channels with exits, i.e., there exist external nets. A left exit or left external net is a net that connects at least one terminal outside the channel. The channel routing phase has determined that the net will exit the channel from its left end only. We define similarly right exits (right external nets) and external nets with two exits. Let m be the number of terminals in the channel. For simplicity, assume that the number of external nets is O(m). (This is not true, however, if the number of external nets with two exits and no terminals on the channel is large. In those cases the time complexity will depend on the number of the latter nets as well). The algorithm in [4] finds an optimal density for channels with two terminal nets in O(m) time. However, the algorithm in [5] guarantees density within one unit from the optimal, for instances with multiterminal nets in O(m \cdot log m) time. Leong [15] also examined the simplified instance of Cong [4], and Cong and Khoo [5] but he only considered channels without exits.

Let T and B be the number of cells at the top and bottom sides of the channel. Throughout this paper we assume that on each channel side the cells are labeled from left to right as c_{1T}, \ldots, c_{iT} and c_{1B}, \ldots, c_{iB}. Let n, m, e, D be the number of terminal positions on either side of the channel, the total number of the terminals which are endpoints of a net, the number of external nets, and the maximum number of terminals in a net, respectively. We assume that e = O(m). We call internal any net which is not external.

In Section 2 we consider Cong and Khoo’s [5] channel routing problem where there exists only one cell per side and we present optimal algorithms that minimize the density in O(m) time. There are three definitions of the density in the literature. The first density definition is given in [15, 17, 3, 25]. The density d of the channel is defined to be equal to the maximum number of nets that can be routed in one layer of the underlying rectilinear grid: The density at a column \( r_1 \), is equal to the maximum number of nets crossing column \( r_1 \), and at least one terminal outside column \( r_1 \). The open density of a channel is equal to the maximum number of nets that intersect with a vertical line between any two columns of the underlying rectilinear grid. This density definition is very useful in the knock knee model [14]. Finally, the closed density, useful in the Manhattan model [14], is equal to the maximum number of nets that have one terminal in a column \( \leq r_1 \), and another in a column \( \geq r_1 \). We give O(m) algorithms that obtain optimal density. We consider all three density definitions. Moreover, we consider the variation proposed in [5] where only one of the two cells has interchangeable terminals, and we show that the density minimization problem is NP-hard for all three density definitions.

In Section 3, we consider the via minimization problem [14, 19], the problem of finding the maximum number of nets that can be routed in one layer [22], and the problem of determining whether the channel adopts internal-external layout [14, 26]. All three problems are studied for the general instance with an arbitrary number of cells with interchangeable terminals on both sides. We present an O(m) time algorithm for the first problem, an O(m^2 \cdot D) time algorithm for the second and an O(m) time algorithm for the third problem. For the maximum planar subset problem we also give an alternative O(m) time algorithm for the special instance of channels that have one cell per side.

\(^1\text{Observe here that the closed density is no less than the first density definition and an algorithm that obtains the latter density guarantees optimal closed density as well.}\)
2. DENSITY MINIMIZATION

We consider the density minimization problem. In Section 2.1 we present linear time optimal algorithms for channels with one cell per side. In Section 2.2 we show that the density minimization problem is NP-hard if we modify the instance so that the terminals on one side are fixed. However, we show that special cases can be solved or approximated efficiently.

2.1 One Cell per Channel Side

The first density definition We first consider the density definition in [3, 15, 17, 25] and we give a simple algorithm, we call it Algorithm 1, that obtains optimal density if there are no external nets. A similar algorithm has been presented in [15]. However, we present Algorithm 1 in detail since we later modify it to handle open and closed density. We are based on the observation that for this special case the terminals can be interchanged so that the density is either one or two. Let \( k_i \) denote the value of net \( N_i \), i.e., the difference between the number of terminals on the top side minus the number of terminals on the bottom side. (Therefore the value of a net can be a negative number.) Algorithm 1, uses as a subroutine \text{Internal}_\text{Nets} which interchanges the terminals so that the density is never more than two.

Procedure \text{Internal}_\text{Nets} has as input parameters three stacks \( S_1, S_2, S_3 \) of nets. The stacks are generated from a procedure, which we call the \text{preprocessing}_\text{step}, as follows:

\begin{verbatim}
procedure \text{preprocessing}_\text{step}
Input: A channel with interchangeable terminals.

(a) Sort, using bucketsort, all the nets according to their values.
(b) Scan the sorted list and push each net in stack \( S_1 \) or \( S_2 \) or \( S_3 \), depending on whether its value is positive, zero or negative, respectively, so that the top of each stack has net \( N_i \) with the smallest absolute value of \( k_i \) (among the remaining nets in the stack).
(c) return \((S_1, S_2, S_3)\)

end of \text{preprocessing}_\text{step}
\end{verbatim}

\begin{verbatim}
procedure \text{Internal}_\text{Nets}
Input: A channel with internal nets only.
Output: A terminal assignment so that the density (as defined in [3, 15, 17, 25]) is no more than two.

(a) Place all the nets of stack \( S_2 \) at the leftmost available terminal positions so that they are aligned;
(b) \( S := S_1 \);
(c) while both \( S_1 \) and \( S_3 \) are not empty do
    begin
        If \( S = S_1 \) then pop a net \( N_i \) from \( S \) and place it on the leftmost positions as illustrated in either Figure 1(a) or 1(b) or 1(c);
        else pop a net \( N_i \) from \( S \) and place it on the leftmost positions as illustrated in either Figure 1(d) or 1(e);
        (Observe that, at each iteration of the if-then-else statement above, only one of the terminal assignment rules of Figure 1 applies. Moreover, the patterns illustrated in Figures 1(c) and 1(e) never apply if our channel instance does not have external nets. As we describe in Algorithm 2 later, procedure \text{Internal}_\text{Nets} is also used when the channel instance
\end{verbatim}

![Figure 1](image)

**FIGURE 1** Illustration of procedure \text{Internal}_\text{Nets}. Placed terminals are represented by solid cycles, the terminals of the most recently placed net are represented by empty cycles.
has external nets as well. In this case, the terminals in stack $S_2$ must be assigned as Figures 1(c) and 1(e) illustrate. The terminals in stack $S_2$ are the shaded terminals and have been assigned so that they are aligned.

If the number of placed terminals on the bottom channel side is less than the number of placed terminals on the bottom side then $S := S_3$ else $S := S_1$.

(d) Let $S$ be the nonempty stack.
Place the terminals of each net $N_i \in S$ at consecutive channel positions so that no two nets $N_i, N_j \in S$ intersect each other. (Such placement is always feasible.)

end of Internal_Nets

Clearly, the channel density is no more than two. At Step (a), we insist that the nets in $S_2$ are aligned. This is important for Algorithm 2 (described later) that handles external nets. Let $m$ be the number of the terminals in the channel. The sorting of the nets in the preprocessing step is done with bucketsorting. The buckets correspond to the values of the nets. Bucketsorting takes $O(m)$ time since for every net $N_i$ we have that $k \in [-\frac{m-1}{2}, \frac{m-1}{2}]$. Steps (a)–(d) of Internal_Nets require $O(m)$ time. Therefore the overall time complexity is $O(m)$.

Algorithm 1

Input: A channel with one cell per side and internal nets only.

Output: A placement of the terminals so that optimal density (as defined in [3, 15, 17, 25]) is obtained.

(a) Place all the nets $N_i$ with $k = 0$ at the leftmost channel positions so that they are aligned;
(b) Transform each net $N_i$ of value $k \geq 1$ to the pattern of Figure 2(a) by adding zero terminals on the bottom side; (These terminals will be part of $N_i$.)
If $\exists$ enough zero terminals then go to Step (e);
(c) Transform each net $N_i$ of value $k \leq 1$ to the pattern of Figure 2(b) by adding zero terminals on the top side. (These terminals will be part of $N_i$.)
If $\exists$ enough zero terminals then go to Step (e);
(d) (Now the density is one.)
Place the transformed nets in Steps (b) and (c) at the leftmost available positions; exit;
(e) (Now the density is two).
$(S_1, S_2, S_3) :=$ preprocessing_step;
call Internal_Nets

end of Algorithm 1

Algorithm 1 obtains optimal density in $O(m)$ time. For channels that do not have external nets the alignment during Step (a) is trivially obtained by simply placing all nets with $k_i = 0$ on the leftmost channel positions. The explicit alignment at Step(a) is important for Algorithm 2, given below, which handles external nets and uses Algorithm 1 as a subroutine.

Algorithm 2 is an improvement of the $O(m \log m)$ time, near optimal algorithm in [5]. W.l.o.g. we assume that the channel instance does not contain external nets with two exits since the terminals of each such net, if any, can be placed anywhere without affecting the channel density. (We consider them as zero terminals.) Let $d_i$ be the channel density when we only consider the external nets with left exits, $d_i$ be the channel density when we only consider the external nets with right exits. Clearly $\max\{d_i, d_j\}$ is a lower bound for the channel density. Since we have exits, the above quantity cannot be zero. Assume w.l.o.g. that $d_i \leq d_j$.

Algorithm 2 first uses procedure preprocessing_step to generate stacks $S_1, S_2$ and $S_3$. Then it calls procedure concatenate($S_a, S_b, S_c, S_d$) which has an input parameters stacks $S_a, S_b$, and $S_c$ and as an output parameter stack $S_d$. The input parameters of concatenate can be any of the stacks $s_i, i \in \{1,2,3\}$. Procedure concatenate generates a new stack $S_d$ which contains, from top to bottom, the items of stack $S_a$ (in the same relative order) then the items of stack $S_b$ (in the same relative order), and finally the items of stack $S_c$ (in the same order). We found it more convenient to denote the operation of procedure concatenate($S_a, S_b, S_c, S_d$) as $S_d := S_a|S_b|S_c$.

![FIGURE 2 Transformations of nets in Algorithm 1. A net with (a) positive, (b) negative value and its representation. Added zero terminals are represented by empty cycles. The terminals are assigned on consecutive positions after the transformation.](image-url)
Before we proceed to the description of Algorithm 2, we illustrate how we obtain optimal density in the special case where there is only one left exit $N_i$ with value $k_i$, i.e., $d_i = 1$ and $d_i = 0$. Assume w.l.o.g. that $k_i > 0$. We use $\{\}$ to denote an empty stack. In this special case, we work as follows:

Place the terminals of $N_i$ at the leftmost positions in the channel;

if $\exists k_i$ zero terminals on the bottom side then begin
Place the $k_i$ zero terminals at the columns where net $N_i$ has an assigned terminal; call Algorithm 1; exit;
end
else begin
(S1, $S_2$, $S_3$) := preprocessing_step; call $S_1 := N_i[S_1]$; call Internal_Nets; exit
end

Observe that if there exist $k_i$ zero terminals on the bottom channel side, the density can be either one or two. In order to guarantee density one the zero terminals must be placed as indicated and then we have to call Algorithm 1. Otherwise the density is two. Notice that in order to obtain density at most two we must place the left exit $N_i$ at the leftmost positions. We assumed that the left exit has more terminals on the top side. Therefore net $N_i$ is placed at the top of stack $S_1$ and is treated as an internal net. Now Internal_Nets guarantees that the density is two.

In a way similar to the algorithm above, one can derive algorithms to handle the following two special cases: (a) $d_i = 0$, $d_i = 1$ and (b) $d_i = d_i = 1$. Therefore we assume that Algorithm 2 below considers only cases where $d_i$ and $d_i$ are at least two. We use $[\cdot]$ to denote an empty stack. Let $a$ be an integer. We use $|a|$ to denote its absolute value.

Algorithm 2
Input: A channel with one cell per side and external nets.
Output: A placement of the terminals so that optimal density (as defined in [3, 15, 17, 25]) is obtained.

(a) Assume w.l.o.g. that the left and the right exits have more terminals on the top channel side; Consider first the left exits. Sort (using bucket-sort) the left exits according to the number of terminals they have on the top side; Push them sorted in stack $S_l$ so that the top of $S_l$ has net $N_i$ with the smallest number of nets on the top channel side; Similarly, create a stack $S_r$ for the external nets with right exits;

(b) repeat Pop a pair of terminals on net(s) in $S_l$ to be assigned on the leftmost available column until there are no terminals on left exits to be assigned to the bottom channel side;

(c) Repeat Step (b) above in a symmetric way for the external nets with right exits. Now some nets in stacks $S_l$ and $S_r$ have been assigned terminals. Steps (d)-(f) below treat the nets (or parts of nets that consist of the unassigned terminals on the nets) in stacks $S_l$ and $S_r$ as internal nets.

(d) Case 1: $d_i > d_i$.
(d1) if $d_i > d_i$ and there exists at least one left exit where all its terminals have been assigned then begin
(S1, $S_2$, $S_3$) := preprocessing_step; $S_1 := S_l[S_l]$; call Internal_Nets; exit
end
(d2) if $d_i > d_i$ and there is no left exit that has all terminals assigned
then begin
(d2.1) Consider the left exit $N_i$ that was placed last and let $K_i$ denote the number of columns with one assigned terminal.
(d2.2) if $\exists$ zero terminals on the bottom side then assign them at the leftmost available positions;
(d2.3) $(S1, S_2, S_3)$ := preprocessing_steps; $S_1 := S_l[S_l]$; call Internal_Nets; exit
end

(e) Case 2: $d_i < d_i$. This case is handled in a way symmetric to Step (d) above.

(f) Case 3: $d_i = d_i$.
(f1) If there exists at least one right exit that has all terminals assigned then work as in Step (d);
(f2) If there exists at least one left exit that has all terminals assigned then work as in Step (e);
else begin (f2.1) Consider the left exit $N_i$ that was placed last and let $K_i$ denote the number of columns with one assigned terminal. Similarly, consider the right exit $N_i$ that was placed last and let $K_i$ denote the number of columns with one assigned terminal.
(f2.2) if $\exists K_i + K_i$ zero terminals on the bottom side then assign them at the $K_i$ leftmost and $K_i$ rightmost positions, respectively;
(f2.3) $(S1, S_2, S_3)$ := preprocessing_step; $S_1 := S_l[S_l]$; call Internal_Nets; exit
end

end of Algorithm 2

Theorem 2.1 Algorithm 2 interchanges the terminals so that a minimum density (as defined in [3, 15, 17, 25]) is obtained in $O(m)$ time.

Proof: The time complexity of the algorithm is $O(m)$ since the sorting of nets at Step (a) and procedure Internal_Nets require $O(m)$ time. Below we
discuss the optimality of Algorithm 2. It is clear that the terminals on left and right exits must be placed at the leftmost and rightmost channel columns, respectively. We show here the optimality of the case described at Step (d) and similar arguments can apply for Steps (e) and (f).

If the condition at Step (d1) is satisfied, our algorithm guarantees that the assignment of the remaining terminals gives overall density \( d \) which is a lower bound to the channel density. This is guaranteed by the simple modification of \( \text{Internal\_Nets} \) described at Step (d1). This step allows procedure \( \text{Internal\_Nets} \) to manipulate some left exits as internal nets. These left exits have terminals on the top side only. Furthermore, they are assigned left of any internal net in the channel. Opposite to each such left exit there will be at most one internal net. However, the condition at Step (d1) guarantees that, before procedure \( \text{Internal\_Nets} \) operates, there is at least one left exit whose all terminals have already been assigned. Therefore the density is at most \( d \).

Assume now that the condition of Step (d2) is satisfied. Here Step (d2.3) guarantees that the density is at most \( d + 1 \). In fact, in order to guarantee density \( d \) we must have \( K_i \) zero terminals on the bottom channel side so that one can enforce that all the terminals of one left exit have already been assigned when Step (d2.3) starts. Thus, it is significant to assign the left exits so that we minimize \( K_i \). Step (a) sorts the external nets so that we obtain minimum value of \( K_i \). This is shown below.

If there is no left exit with a terminal on the bottom side then the value of each left exit equals to the number of its terminals that must be assigned on the bottom side and our sorting manifestly guarantees minimization of \( K_i \). Assume now that there is at least one left exit that connects a terminal on the bottom side. Let \( B \) be the sum of the terminals on left exits that have to be assigned on the bottom channel side. Clearly, if there is a left external net \( N_i \) that has \( B + K_i \) terminals on the top channel side, then it suffices to have \( K_i \) zero terminals on the bottom side to guarantee density \( d \). Since we sort the external nets according to the number of terms that they have on the top channel side, the external net at the top of the stack guarantees minimization of \( K_i \). In Figure 3 we illustrate that the sorting at Step (a) minimizes \( K_i \). The terminal assignment in Figure 3(a) gives \( K_i = 1 \) while the assignment in Figure 3(b) gives \( K_i = 2 \) and is suboptimal.

**Closed and open density minimization** Algorithms 1 and 2 serve for minimizing the closed density as well. (In fact, the closed density is never less than the density definition that we examined previously.) However, we need to modify Algorithms 1 and 2 in order to minimize the open density. The modified algorithms are called Algorithm 1a and Algorithm 2a, respectively. Procedure \( \text{Internal\_Nets} \) and its **preprocessing\_step** are as in Algorithm 1.

**Algorithm 1a**

**Input:** A channel with one cell per side and internal nets only.

**Output:** A placement of the terminals so that optimal open density is obtained.

(a) Place all the nets with \( k_i = 0 \) at the leftmost channel positions so that they are aligned;
(b) Transform each net \( N_i \) of value \( k_i > 2 \) to the pattern of Figure 4(a) by adding zero terminals on the bottom side; (These terminals will be part of \( N_i \).)
(c) Transform each net \( N_i \) of value \( k_i < -2 \) to the pattern of Figure 4(b) by adding zero terminals on the top side. (These terminals will be part of \( N_i \).)
(d) (In this step we attempt to obtain density one.) Let \( r_1, b_1, r_2, b_2 \) be the set of nets \( N_i \) with \( k_i \in \{1, -1, 2, -2\} \). Let \( |r_1|, |b_1|, |r_2|, |b_2| \) be the number of nets in each set.

(d1) If \( |b_2| > |r_2| + 1 \) then combine the nets \( N_i \in b_2 \cap t_2 \) to form patterns as in Figure 4(c); We call each such pattern an island. Island 1 is formed by combining \( |t_2| + 1 \) nets in \( b_2 \) with the nets in \( t_2 \). Each of the remaining islands is a net \( N_i \in b_2 \) which is not in island 1;

Add zero terminals or nets \( N_i \) with \( k_i = 1 \) before and after each island as Figure 4(d) illustrates; (We always have enough nets or zero terminals to do these island transformations.)

Place the transformed islands one after the other at the leftmost available positions; Place all the remaining nets \( N_i \) with \( k_i \in \{1, -1\} \) so that the density remains 1; (Figure 4(e) illustrates such a placement for the case when we

![Figure 3](image-url)
FIGURE 4 Transformations on Algorithm 1a. (a) A net with positive value and its transformation. Added zero terminals are
represented by empty cycles. (b) A net with negative value and its transformation. Added zero terminals are represented by empty
cycles. (c) Three islands: An instance where $|b_1| = |s| + 3$. (d) Island 2 of part (c) above is combined with one zero terminal and a
net with value 1. (e) Illustrating the transformation. (f) $|k| = |b_3| + 1$. (g) Two zero terminals are added on the island of part (f).
are left with two nets $N_i$ with $k_i = 1$, one net $N_j$ with $k_j = -1$ and a zero terminal on the bottom side.)

(d2) if $|b_2| < |r_2| + 1$ then work in a way similar to Step (d1);
(d3) if $|b_2| = |r_2| + i, i \in \{-1, 0, 1\}$, then form only one island as in Figure 4(f) (or its mirror image);
Transform the island as Figure 4(g) illustrates by adding either zero terminals or nets $N_i$ of value $k_i \in \{1, -1\}$;
Place the transformed island at the leftmost available channel positions;
Place the remaining nets as in Step (d1); (See also Figure 4(e));
(d4) exit;
(e) (Now the density is two. Ignore Steps (a)-(c) of the algorithm.)
(S1, S2, S3) := preprocessing_step;
call Internal_Nets
end of Algorithm 1a

Theorem 2.2 Algorithm 1 obtains optimal density in $O(m)$ time.

Proof: The optimality of the algorithm is justified by proving that Steps (a)-(d) correctly determine whether the channel density is one or two. Clearly, Step (a) does not affect the optimal solution, since any optimal solution can be easily transformed to an instance that satisfies the placement described at Step (a) without an increase in the density.

Observe that in order for the density to remain one, any net $N_i$ with value $-2 > k_i > 2$ needs to have additional zero terminals on the opposite channel side, as described in Steps (b) and (c). If these zero terminals do not exist, the density is two. We then generate the patterns illustrated in Figures 4(a) and 4(b) since these patterns allow us to place net $N_i$ with $k_i = 2$ next to net $N_j$ with $k_j = -2$ so that we maintain density one without placing any zero terminal between the two nets. In Figures 5(a)–5(d), we illustrate that if we do not follow the patterns described in Steps (b)–(c), the density will always be optimal.

We claim now that the density is one after all nets $N_i$ with $-2 \geq k_i \geq 2$ have been successfully transformed to one of the patterns of Figures 4(a) and 4(b). In Step (d) we describe a placement that obtains optimal density. First we combine the nets $N_i$ with $k_i \in \{2, -2\}$ so that they share columns as much as possible. (For example, Figure 5(d) shows that we need two additional zero terminals if the described condition is not satisfied.) If we succeed to a placement that aligns the terminals of all nets $N_i$ with $k_i \in \{1, -1\}$ as Figure 4 shows, the latter nets are identical to zero terminals and the density will be one. But after Steps (b) and (c) such a placement is always feasible.

For the time complexity analysis observe that Step (a) takes $O(m)$. In $O(m)$ time we are also able to determine the number of zero terminals on each channel side. This ensures that Steps (b) and (c)
require $O(m)$ time. Similarly Step (d) requires $O(m)$ time.

Algorithm 2a is now the same as Algorithm 2 but now in the special case where $\max\{d_1, d_2\} \leq 1$ we substitute $k_i$ by $k_i - 1$. Similarly, at Steps (d)-(f) of Algorithm 2, we must substitute $K_i$ and $K_r$ by $K_i - 1$ and $K_r - 1$, respectively. The proof of correctness is similar to the one in Algorithm 2 and is omitted.

2.2 Fixed Terminals on One Side

We now examine a variation of the channel density minimization problem considered by Cong and Khoo [5]. The channel consists of two cells, one cell per channel side, the terminals on one cell (w.l.o.g. the bottom cell) are fixed, and the terminals on the other cell are interchangeable. We initially show that the problem of obtaining minimum channel density by interchanging the terminals in the top channel cell is NP-hard. This is shown for all three density definitions, i.e., the density definition by Cai and Wong [3] and the open and closed density definitions [14]. We prove the latter by reducing the bin packing problem, defined below, to the decision version of our problem in which we are examining whether there exists a terminal permutation such that the channel density is less than or equal to an input integer $K$.

NP-complete proofs Our NP-complete reductions are done from the bin packing problem, which is NP-complete in the strong sense [8], and is defined below.

Bin packing instance: A finite set $U$ of items, a size $s(u) \in \mathbb{Z}^+$ for each $u \in U$, a positive integer bin capacity $B$ and a positive integer $M$.

Question: Is there a partition of $U$ into disjoint sets $U_1, U_2, \ldots, U_M$ such that sum of the sizes of the items in each $U_i$, $1 \leq i \leq M$, is $B$ or less?

Theorem 2.3 The $1-DD$ problem is NP-complete.

Proof: Clearly, $1-DD$ is in NP. Consider an instance of the bin packing problem as described above. Let $|U|$, $|U_i|$ and $N_i$ denote the size (number of items) of the set $U$, the size of each set $U_i$ $1 \leq i \leq M$, and the number of items $u \in U$ with $s(u) = 1$, respectively. We construct the channel instance for the $1-DD$ problem as follows:

The length, $n$, of the constructed channel is equal to $M \cdot B + 2 \cdot (M - 1)$. The top side of the channel has $N_i$ zero terminals, one for each item $u \in U$ with $s(u) = 1$. We call each such terminal an item zero terminal. We have $|U| - N_i$ nets, each having terminals in cell $C$. Each one of these nets corresponds to an item $u \in U$ with $s(u) \geq 2$. We call such a net an item net. For each item $u$ with $s(u) \geq 2$ there exists an item net connecting $s(u)$ terminals. The remaining terminals on the top side are zero terminals. See also Figure 6.

The bottom channel side consists of $M - 1$ two terminal nets labeled $\forall i$, $1 \leq i \leq M - 1$. Net $\forall i$ has terminals at positions $2 \cdot (B + 2) - 1$ and $2 \cdot (B + 2)$. The remaining terminals on the bottom side are zero terminals. We call these zero terminals hole zero terminals since they do not contribute to the channel density. Any $B$ consecutive hole zero terminals form a hole $c_i$, $1 \leq i \leq M$, which corresponds to set $U_i$ of the bin packing instance. Clearly, from the description of the top channel side we conclude that the nets in the channel connect terminals either on the top or the bottom side but not on both sides. See also Figure 6.
The number $K$ in the $1-\text{DD}$ instance is equal to 1. The above construction can be done in polynomial time since the bin packing problem is NP-complete in the strong sense and we have selected the sizes of the sets to be polynomial. We will show that the bin packing instance is satisfied if and only if the constructed $DD$ instance is satisfied.

From the channel instance construction, a lower bound to the density is equal to 1. This is encountered by considering the fixed nets on the bottom side only. If the bin packing problem is solvable, then we show that we can always place the item nets opposite to the holes so that no terminal in an item net is placed on a column that has a terminal on a two terminal net on the bottom side. (If a terminal of an item net is placed in such a terminal position, the density becomes 2 and the $1-\text{DD}$ instance is not satisfied.) The placement of the item nets and the item zero terminals is directed by the placement of the respective items $u$ in the bin packing solution. Let the terminal positions of hole $c_j$, $1 \leq j \leq m$, be $B_{c_1}, \ldots, B_{c_{m-1}}$. If the bin packing instance has $r$ items $u_i, 1 \leq i \leq r$, in set $U_i$, the respective item nets and the item zero terminals are placed at positions $T_{u_i}$ up to (at most) position $T_{u_i+r}$, and in the order they are placed on set $U_i$. (This is always feasible). The above guarantees density 1.

Conversely, assume that the density of the channel instance is 1 for some terminal assignment. Let $\Gamma$ be the set of columns that have an item net on a column with a terminal on a two terminal net on the bottom channel side. On each $\tau_i \in \Gamma$ there exists either an item zero terminal or some other zero terminal. Now interchange the terminals so that on each $\tau_i \in \Gamma$ we have zero terminals which are not item zero terminals. Such an interchange is always feasible because we have at least $2 \cdot (M - 1)$ such zero terminals. Now interchange the terminals so that each item net occupies consecutive positions and such that the density is still $B$. It is not difficult to observe that the terminal positions of the item nets and the item zero terminals leads us to an assignment of the corresponding items to set $U_i$. (The description of this assignment is similar to the inverse part of the proof and is omitted.) Thus the bin packing problem is satisfiable.

**Theorem 2.4** The $2-\text{DD}$ problem is NP-complete.

**Proof:** The reduction will be again done from the bin packing problem. The length of the channel is now $2 \cdot (B - 1) + (M - 2) \cdot (M - 2) + 2 \cdot (M - 1)$. The top side of the constructed channel is as in Theorem 2.3. The bottom side of the channel is as in Figure 7. Now, all holes have size $B - 2$ besides the leftmost and rightmost holes that have size $B - 1$. The reason for the latter modification is due to the definition of the open density. Let the terminal positions of hole $c_i$ be in the range $[B_{c_i}, B_{c_i}]$. Then according to the definition of open density we can place nets in the range $[T_{u_i,1}, T_{u_i+1}]$ without increasing the density lower bound due to the bottom channel side. Note that positions $T_{a-1}$ and $B_{a+1}$ are not defined, however. Therefore these two holes have one more terminal position, i.e., $B - 1$ positions. The number $K$ in the $2-\text{DD}$ instance is 1.

If the bin packing problem is satisfiable then the channel density is 1. (The argument is the same as in Theorem 2.3 since we now consider the definition of open density.) Inversely, the sizes of the holes allow us to claim that if the density of the $2-\text{DD}$ instance is 1 then the bin packing problem is satisfiable.

**Theorem 2.5** The $3-\text{DD}$ problem is NP-complete.

**Proof:** The reduction is identical to the one in Theorem 2.3. From the construction of the channel we have that all nets connect terminals on the same side. Observe that for such a channel instance the definition of open density is identical to the density definition in [3].

**Special cases** Although the $i-\text{DD}$, $i \in \{1, 2, 3\}$, is NP-complete, special cases of the optimization problem...
version can be solved optimally or approximated in polynomial time. The algorithms are described for the density definition in [3] but they can be extended to the other two definitions. They can also be extended to handle channels with more than one cell on the top channel side.

**Theorem 2.6** If the channel instance consists of two terminal nets then an optimal density can be found in $O(m)$ time.

**Proof:** We describe a simple algorithm that consists of two steps and satisfies the lemma. In the first step, we align all nets with terminals on both sides. Let $\mathcal{N}$ be the set of nets with both endpoints on the top side. We use $lb$ to denote the density lower bound formed when considering nets with both endpoints on the bottom channel side. In the second step of the algorithm, we determine whether all the remaining two terminal nets with movable terminals can be assigned so that the density lower bound $lb$ is maintained.

Let $m$ be the number of terminals on the channel instance. The alignment step requires $O(m)$ time. (See also [18].) We can show that an optimal solution can be transformed to an instance where all nets are aligned without increasing the channel density. (This is independent of the density definition.) The second step of the algorithm is simple. Let $P$ be the set of columns where maximum density is encountered when we consider the nets with both terminals on the bottom side only. $P$ partitions the terminal positions on the top side and net in $P$ into $s$ sets $S_i$. (Assume w.l.o.g. that the labels are assigned from the to right.) Let $|S_i|$ denote the number of terminal positions $\in S_i$, $1 \leq i \leq s$. If the movable terminals on each net can be placed on the terminal positions of some set $S_i$, $1 \leq i \leq s$, then the optimal channel density is equal to the density lower bound; otherwise the optimal channel density is equal to the density lower bound plus one. We give a simple algorithm that determines whether the density, $d$ is $lb$ or $lb + 1$.

For each net $N_j \in \mathcal{N}_b$, let $r(N_j)$ denote the rightmost terminal position on the bottom side, $l(N_j)$ be the leftmost terminal position on the bottom side and $T(N_j)$ be the number of terminals on the top side. Let $n$ be the channel length.

**Theorem 2.7** If for every pair of nets $(N_i, N_j)$ such that $N_i$ and $N_j \in \mathcal{N}_b$, we have that $r(N_i) < r(N_j)$, $l(N_i) < l(N_j)$ and $T(N_i) = T(N_j) = T$, then we can obtain channel density which does not exceed the optimal by more than one unit in polynomial time.

**Proof:** Clearly, the relative order of the movable terminals of each net $N_j \in \mathcal{N}_b$, if any, does not affect the density. From now on, we consider the channel problem where we have disregarded all nets that have all their terminals on the top side. The density of the induced channel is no less than one unit from the density of our original channel routing problem since we can always place the terminals of the removed nets so that no two such nets intersect with each other.

We now claim that there exists an optimal solution where for any two nets $N_i, N_j \in \mathcal{N}_b$ for which $r(N_i) < r(N_j)$, and thus $l(N_i) < l(N_j)$, all the movable terminals of net $N_i$ are assigned to the left of the movable terminals on net $N_j$. This is efficient for our approximation algorithm since we have now fixed the relative order of all the movable terminals on nets in $\mathcal{N}_b$ and we can use the algorithm in [3] to solve the latter problem in $O(n^2)$ time or the algorithm in [10] in $O(n \log n)$ time.

We show the claim as follows: Assume that there exists an optimal assignment of the terminals that do not follow the ordering of our claim. We show a series of terminal interchanges which does not increase the density and results to an assignment as in our claim.

Label the nets in $\mathcal{N}_b$ so that if $r(N) < r(N_i)$ then $i < j$. Now the nets in $\mathcal{N}_b$ are labeled $N_i$ to $N_j$. Assume also, for easier description, that the nets satisfy that $r(N_i) < l(N_i)$. The bottom channel side looks as in Figure 8(a). For the remaining of this proof whenever we refer to a net $N_i$ we imply that $n_i$ is in $\mathcal{N}_b$.

Consider first nets $N_i$ and $N_{i-1}$ in the optimal solution. If there is one movable terminal of $N_i$ left to a movable terminal of $N_{i-1}$ then interchange their positions. At the end of this step all movable terminals of $N_i$ are to the right of the movable terminals of $N_{i-1}$. This interchange does not increase the density at columns right of $r(N_{i-1})$. Observe that both nets $N_i$ and $N_{i-1}$ have $T$ movable terminals. Therefore, the interchange does not also increase the density left of $r(N_{i-1})$.

The latter is not true if the nets do not have the same number of movable terminals. For example, in
Figure 8(b) net $N_3$ has 5 movable terminals and net $N_5$ has two movable terminals. The figure shows the optimal terminal assignment which clearly does not follow the ordering in the claim. If we interchange the movable terminals of nets $N_3$ and $N_5$ as described above, we will still have one movable terminal of net $N_3$ right at the column $r$ at position $r(N_3)$. This interchange will then increase the density at column $r$ and therefore the overall channel density.

Now we finished interchanging terminals on nets $N_r$ and $N_{r-1}$. This terminal interchange is then repeated for nets $N_r$ and $N_{r-2}$, then nets $N_r$ and $N_{r-3}$ and so on until we cannot interchange the position of a movable terminal from net $N_r$. Then we repeat the above terminal interchange procedure by just decrementing the subscripts of the nets involved previously, i.e., we first consider nets $N_{r-1}$ and $N_{r-2}$. This concludes the proof of the lemma.

3. OTHER OBJECTIVE FUNCTIONS

In this section we give polynomial time algorithms that find optimal solutions for the via minimization problem, the problem of finding the maximum number of nets that can be routed on a single layer, and to determine whether the channel adopts internal-external layout. The input instance in all three problems is a channel with an arbitrary number of cells on each side.

3.1 Via Minimization

We present an algorithm, we call it via_minimization, that minimizes the number of via holes needed for the routing of a channel. Any optimal solution for this problem requires that each net consists of at most one trunk section [14, 19]. If this was not the case and there exists an optimal solution with net $N_j$ having more than trunk, we can reduce the number of trunks of $N_j$ to be one and thus reduce the number of vias; a contradiction to the assumption that we had an optimal solution. We also assume that we do not have any external nets with two exits and no terminals in the channel since none of these nets requires any vias.

Observe also that an alignment of any two terminals of a multiterminal (internal or external) net or an external two terminal net reduces the number of vias by one, but an alignment of the two terminals of a two terminal net reduces the number of vias by two. Thus the problem of minimizing the number of vias is equal to the problem of maximizing the quantity $2 \cdot A_{tr} + A_{ex}$, where $A_{tr}$ is the number of aligned two terminal internal nets and $A_{ex}$ is the number of aligned terminal pairs on any other net. This is a weighted version of the maximum alignment problem that Lin and Sahni examined in [18] for internal nets only. Now the alignment of the two terminals of an internal two terminal net equals to the alignment of two pair of terminals on any other net. Lin and Sahni’s algorithm [18], we call it Lin_Sahni(C), does not obtain optimal solution to our problem even for a channel $C$ with internal nets only. We give a counterexample of the latter in Figure 9. In Figure 9(b) we construct the bipartite graph for the channel in Figure 9(a) as Lin and Sahni suggested [18]. In this graph there exists an edge between any pair of terminals that are in an overlap region [18] and on the same net. (The reader is referred to [18] for formal definitions and description of the algorithm.) Then based on the graph of Figure 9(b), the resulting terminal interchange is shown in Figure 9(c). This is a suboptimal solution to our problem since the total number of vias for this interchange is seven but the minimum number of vias is six.

We show now that Lin and Sahni’s algorithm [18] can serve as a subroutine for a simple algorithm that does not obtain via minimization. First, we give a straightforward modification of algorithm Lin_Sahni(C), we call it generalized_Lin_Sahni(C), to obtain maximum alignment of a channel instance $C$ with internal and external nets. (We will omit the
proof of optimality for the later algorithm.) We also assume that each external net has at least one terminal on some side of the channel; otherwise it requires no via and it is omitted. Then we present algorithm \textit{via\_minimization} (Algorithm 3) that uses \textit{generalized\_Lin\_Sahni}(C) as a subroutine.

\textbf{procedure} \textit{generalized\_Lin\_Sahni}(C)

\textbf{Input:} A channel C with internal and external nets.
\textbf{Output:} An interchange of the terminals that obtains maximum alignment.

Set \( J := n + 1; \)

\textbf{while} \( \exists \) an external net \( N_i \) \textbf{do begin}

Place a new terminal \( t_i \) at position \( T_i \) and in a new cell, \( j := j + 1 \)

\textbf{end}

Let \( C_1 \) be the new channel; call \textit{Lin\_Sahni}(C)

\textbf{end of} \textit{generalized\_Lin\_Sahni}(C)

\textbf{Algorithm 3 (via\_minimization)}

\textbf{Input:} A channel C with internal and external nets.
\textbf{Output:} A terminal interchange that obtains minimum number of vias.

\begin{enumerate}
\item[(a)] Obtain maximum alignment in the reduced channel consisting of the two terminal internal nets only;
\item[(b)] Contract the columns that have aligned nets; Let \( C_2 \) be the reduced channel instance;
\item[(c)] call \textit{generalized\_Lin\_Sahni}(C)
\end{enumerate}

\textbf{end of Algorithm 3}

\begin{theorem}
Algorithm 3 obtains via minimization in \( O(m) \) time.
\end{theorem}

\textbf{Proof:} The optimality of the algorithm is based on the fact that the maximum alignment at Step (c) does not affect the maximum alignment of the remaining terminals since a contracted column with an aligned two terminal net can not be used for aligning more than one net. Moreover, the contracted column has a net with maximum weight. The time complexity is determined by Steps (a) and (c). Step (a) can be easily implemented in \( O(m_1) \), where \( m_2 \) is the number of two terminal internal nets with endpoints on different sides. A greedy left to right alignment of the two terminal nets is optimal in this case. Although the complexity of the algorithm in [18] was mentioned to be \( O(n) \), it is indeed only \( O(m) \). Thus the overall time complexity of Algorithm 3 is \( O(m) \).

3.2 Maximum Planar Subset of Nets

Our algorithm (Algorithm 4) for the maximum planar subset of nets consists of three phases. The first phase (phase-I) transforms the input channel instance to one consisting of internal nets only as follows. The channel has two more cells \( c_{z} \) and \( c_{r} \) at the leftmost and rightmost positions on the (w.l.o.g.) top side. Each external net with a left exit has an additional terminal on \( c_{z} \) and each external net with a right exit has an additional terminal on \( c_{r} \). For simplicity, assume that the number of terminals in the new channel is \( m \). The second phase of the algorithm interchanges the terminals on their cells so that we avoid the intersection between any two nets \( N_i \) and \( N_j \) if this is possible. This is procedure phase\_2. Finally, the last part of the algorithm selects the maximum planar subset of nets.

We first describe procedure phase\_2. The procedure ignores all nets that connect terminals in the same cell since we can avoid the intersection of such a net with any other net. Also assumes that each net has at most one terminal per cell. If this is not the case, the terminals of each net in a cell will be assigned consecutive positions. Finally, for the simplicity of the description, assume that there are no two nets, each connecting exactly two terminals one in cell \( c_{z} \) and the other in cell \( c_{r} \). Phase\_2 consists of two steps and is given below.
Step (a) consists of a series of transformations in the channel. We first transform the channel to a switchbox $C$ as shown in Figure 10. (Here, each vertical side is a cell that has zero terminals only.) We relabel the cells in the channel from $c_1$ to $c_6$, by arbitrarily selecting a cell and moving in a clockwise manner. Similarly relabel in a clockwise manner the terminal positions in the channel. We now define, for each net, the first, last and middle terminals, subject to the clockwise ordering that we imposed. We also define for each cell the low and the high boundary, subject to our ordering. See also Figure 10.

Step (b) assigns the terminals of the nets in their cells as follows. We scan the cells in clockwise manner starting from cell $c_1$. For each cell $c_i$, we assign terminal $t_k$ of net $N_k$ as follows: If it is the first terminal it is assigned next to the high boundary, if it is the last terminal it is assigned next to the low boundary and if it is a middle terminal it is assigned anywhere in cell $c_i$.

Furthermore, we determine the relative order of any two first terminals $t_k$ and $t_i$, on nets $N_k$ and $N_i$, respectively, as follows: If the last terminal of net $N_k$ is on a cell that has higher label than the one where the last terminal of $N_i$ resides, terminal $t_k$ will be assigned closer to the high boundary of cell $c_i$ than terminal $t_i$. If the last terminals of nets $N_k$ and $N_i$ are on the same cell and one of the two nets is a two terminal net, w.l.o.g., net $N_i$, terminal $t_i$ is assigned closer to the low boundary of cell $c_i$. This concludes the description of procedure phase_2.

Step (b) can be implemented in $O(m)$ time using bucketsorting as in [25]. We skip here these implementation details. In Figure 11 we show how Step (b) succeeds in placing the terminals in their cells so that we avoid the intersection between any two nets. (In the figure we assume that phase_2 has assigned the terminals.) Observe that in cell $c_1$ net $N_2$ does not intersect with net $N_1$. We avoided such an intersection by assigning the terminal on net $N_2$ in cell $c_1$ to the right of the terminal on net $N_1$ and the terminal on net $N_2$ in cell $c_{i-1}$ to the left of the terminal on net $N_1$. Observe that both nets $N_1$ and $N_2$ have their last terminal on cell $c_i$, and they have a terminal in cell $c_{i-1}$. The relative order of the terminals in cell $c_{i-1}$ is determined by the fact that terminal on net $N_2$ in that cell is not the last one.

Similar arguments hold when we consider any two nets in Figure 11. For example, the assignment guarantees that net $N_2$ will not intersect with all the other
nets. Observe that although nets $N_3$ and $N_4$ do not intersect with net $N_1$ and $N_2$, we can not avoid the intersection between nets $N_3$ and $N_4$. This is the reason that Step (b) of phase$_2$ allows the relative positions of the terminals on nets $N_1$ and $N_2$ to be arbitrary on cell $c$. From the above discussion we have:

**Lemma 3.1** Procedure phase$_2$ interchanges the terminals in $O(m)$ time so that we avoid the intersection between any two nets.

We describe now a dynamic programming approach (last phase of the algorithm) that obtains the minimum planar set of nets in the switchbox $C$. This is an extension of Supowit’s algorithm [22] to multiterminal nets. From $C$ construct a circle graph $G = (V, E)$ so that there exists a vertex for each net and an edge $e \in E$ between vertices $v$ and $w$ iff net $v$ intersects with net $w$. We also assume a numbering of the terminals in $C$ from 0 to $m - 1$. Observe that our problem is equivalent to finding a maximum independent set $I(0, m - 1)$ on graph $G$.

In order to find the maximum independent set on graph $G$ we must first find the maximum independent set $I(i, j)$ for each circle graph $G_{i,j} = (V_{i,j}, E_{i,j})$ which is induced from graph $G$ as follows: $V_{i,j}$ consists of all nets with all terminals having labels in the range $[i, j]$. Edge $e = (v, w) \in E_{i,j}$ iff $v, w \in V_{i,j}$ and net $v$ intersects with net $w$. We maintain a two dimensional array $I$. Entrance $I(i, j)$ will have the maximum independent set of graph $G_{i,j}$. $I(i, j)$ is computed before $I(i, j_2)$ if $j < j_2$. Observe that in order to find the maximum independent set for graph $G$ we must find $I(0, m - 1)$ since $G_{0,m-1} = G$. To compute $I(i, j)$, we let $k_1, k_2, \ldots, k_r, j$ be the terminals of net $N_i$ where terminal $j$ is on. If at least one of the $k_i$ terminals, $1 \leq i \leq t$, is not in the range $[i, j]$ then $G_{i,j} = G_{i,j-1}$ and we have $I(i, j) = I(i, j - 1)$. If all $k_i$, $1 \leq i \leq t$, are in the range $[i, j]$ (now net $N_i$ corresponds to a vertex $v_j \in G_{i,j}$) then we distinguish between two cases: if $v_j \in I(i, j)$ then by definition of an independent set, $I(i, j)$ contains no vertex $v$, that corresponds to a net $N_i$ with a terminal in $[i, k_1 - 1], [k_1 + 1, k_2 - 1], [k_2 + 1, k_3 - 1], \ldots, [k_t + 1, j]$. Then we set $I(i, j) := I(i, k_1 - 1) \cup I(k_1 + 1, k_2 - 1) \cup I(k_2 + 1, k_3 - 1) \cup \ldots \cup I(k_t + 1, j) \cup \{v_j\}$. However, if $v_j \notin I(i, j)$ then we have $I(i, j) = I(i, j - 1)$ since $G_{i,j}$ and $G_{i,j-1}$ can (in the worst case) differ by vertex $v_j$ and its associated edges. Therefore we set $I(i, j)$ to be the larger of the two sets $I(i, j - 1)$ and $I(i, k_1 - 1) \cup I(k_1 + 1, k_2 - 1) \cup I(k_2 + 1, k_3 - 1) \cup \ldots \cup I(k_t + 1, j) \cup \{v_j\}$.

We summarize all the above in Algorithm 4 below. (We give in details only the third phase of the algorithm.)

**Algorithm 4**

**Input:** A channel with interchangeable terminals.

**Output:** A maximum subset of nets that can be routed on a single layer.

(a) call phase$_1$ and phase$_2$ to construct the circle graph $G = (V, E)$ on which the remaining of the algorithm operates;

(b) for $j := 0$ to $M-1$ do

for $i := 0$ to $M-1$ do

if all the terminals $\{k_1, \ldots, k_r, j\}$ of net $N_i$ are in the range $[i, j]$ and $|I(i, k_1 - 1)| + |I(k_1 + 1, k_2 - 1)| + |I(k_2 + 1, k_3 - 1)| + \ldots + |I(k_t + 1, j)| + 1 > |I(i, j - 1)|$ then $I(i, j) := I(i, k_1 - 1) \cup I(k_1 + 1, k_2 - 1) \cup I(k_2 + 1, k_3 - 1)$

else $I(i, j) := I(i, j - 1)$

end of Algorithm 4

Let $m$ be the number of terminals, and $D$ be the maximum number of terminals on a net. 

**Theorem 3.2** Algorithm 4 finds a maximum planar subset of nets of an arbitrary channel instance with interchangeable terminals in $O(m^2 \cdot D)$ time.

**Proof:** The proof of correctness follows from the above discussion. The time complexity is determined by Step (b), i.e., the third phase of the algorithm. In order to guarantee that the time complexity is $O(m^2 \cdot D)$ we must avoid storing explicitly the elements of $I(i, j)$ when the latter is computed in the body of the interior loop. (If we store the elements explicitly, the time complexity is $O(m^3)$.) We can instead have pointers to the previous entrances in the array. These pointers form a tree structure and at the end of the algorithm we can backtrack on the tree to find the actual nets in the maximum independent set. 

On the remaining portion of this section we focus on the special case where the channel consists of two cells with interchangeable terminals, one cell per side. If the channel has internal nets only then there exists a (trivial) interchange of the terminals so that we can avoid the intersection between any two internal nets in $O(m)$ time. Therefore w.l.o.g. we assume that the channel has external nets. 

For this special case we present a faster algorithm, Algorithm 4a, that finds a maximum planar subset
of the nets in \(O(m)\) time. We can again assume w.l.o.g. that every net (either internal or external) has at most one terminal on each cell, i.e., at most one terminal on each side. Therefore an internal net can now be represented either as a two terminal (TT) net with endpoints in different sides or by a single terminal (ST). We also partition the left exits into three sets: left-exit top-side (LETS) external nets that have a terminal on the top side only, left-exit bottom-side (LEBS) external nets that have a terminal on the bottom side only, and left-exit top and bottom side (LETBS) external nets that have terminals on both sides. We similarly partition the right exits in right-exit top-side (RETS) external nets, right-exit bottom-side (REBS) external nets, and right-exit top and bottom side (RETBS) external nets that have terminals on both sides. Finally, we partition the external nets with two exits in: two-exit no-side (TENS) external nets that they connect no terminal in the channel, two-exit top-side (TETS) external nets that have a terminal on the top side, two-exit bottom-side (TEBS) external nets with a terminal on the bottom side, and two-exit top-bottom side (TETBS) with a terminal on both sides.

Let \(|P_I|\) denote the number of nets in set \(P_I\), \(I \in \{TT, ST, LETS, LEBS, RETS, REBS, TETS, TEBS, TENS, TETBS\}\). We call basic all the nets \(N \in \{ST, LETS, LEBS, RETS, REBS\}\). Observe that all the basic nets can be routed independently of whether there exist additional nets that are not basic. Also observe that the set of nets \(\{TENS, TETS, TETBS, TETBS\}\) are mutually exclusive in the sense that if a network on one set is selected to be part of \(P\), this implies that only one net \(N \in RETB\) will be in \(P\), provided that set \(RETB\) is not empty. 

\[
|P| = \max(|TT| + \min(|RETBS|, 1) + \min(|LETS|, 1), |TENS| + \max(|TEBS|, 1) + \min(|TETS|, 1), |TETBS|) \quad (\text{if, for example, } \min(|RETBS|, 1) \text{ is selected to be part of } P, \text{ this implies that only one net } N \in RETB \text{ will be in } P, \text{ provided that set } RETB \text{ is not empty.})
\]

(b) \(P = P \cup B\)

End of Algorithm 4a

**Theorem 3.3** Algorithm 4a finds a maximum planar subset of nets in \(O(m)\) time.

### 3.3 Internal-External Layout

In this Section, we modify Algorithm 4 of the previous section to determine whether the channel admits internal-external layout. In this decision problem, we want to determine whether the set of nets \(\mathcal{N}_I\) can be partitioned in two sets \(\mathcal{N}_{	ext{in}}\) and \(\mathcal{N}_{	ext{out}}\) so that all nets in \(\mathcal{N}_{	ext{in}}\), \(\mathcal{N} \in \{TT, ST, LETS, LEBS, RETS, REBS\}\), can be routed on the same layer. The new algorithm, we call it Algorithm 5, is given below:

**Algorithm 5**

**Input:** A channel instance with interchangeable terminals on their cells.

**Output:** Determining whether the channel admits internal-external layout.

(a) Use phase_1 of Algorithm 4 to transform all external nets to internal. Then use phase_2 of Algorithm 4 to assign the terminals so that we avoid an intersection between any two nets in the channel instance. Now the cells are labeled from \(c_i\) to \(c_i\), and the input channel has been transformed to a switchbox as in Figure 10.

(b) \(\mathcal{N}_{\text{in}} := \mathcal{N}_I; \mathcal{N}_{\text{out}} := \emptyset; c_0 := c_i;\)

(c) while the switchbox has not been scanned do 

Scan the switchbox in a clockwise manner starting from cell \(c_i\) until you find the first net \(N_i \in \mathcal{N}_{\text{in}}\) and in cell \(c_i\) that intersects with some other net in \(\mathcal{N}_{\text{out}}\):

\(\mathcal{N}_{\text{in}} := \mathcal{N}_I - N_i; \mathcal{N}_{\text{out}} := \mathcal{N}_{\text{out}} \cup N_i; c_0 := c_i;\)

end

(d) call phase_2 of Algorithm 4 to assign the terminals in the induced switchbox instance consisting of the nets in \(\mathcal{N}_{\text{in}}\);

(e) if \(\exists\) nets in \(\mathcal{N}_{\text{out}}\) that intersect with each other then the channel does not adopt internal-external layout else it does adopt.

end of Algorithm 5

**Theorem 3.4** Algorithm 5 determines whether the input channel adopts internal-external layout in \(O(m)\) time.

**Proof:** Algorithm 5 first interchanges the terminals so that we avoid the intersection of any two nets in \(\mathcal{N}_I\). The set of nets \(\mathcal{N}_I\) forms the internal layout and therefore must be planar. Algorithm 5 removes greedily (Step (c)) each net in \(\mathcal{N}_I\) that violates the planarity of set \(\mathcal{N}_I\), and places this net in set \(\mathcal{N}_{\text{out}}\), which forms the external layout and must also be planar.
At Steps (d) and (e), Algorithm 5 determines whether set $\mathcal{N}_a$ is planar. Clearly, if set $\mathcal{N}_a$ is planar then the input channel adopts internal-external layout. Conversely, assume that two nets $N_i$ and $N_j$ in $\mathcal{N}_a$ intersect with each other. Then, in order to have internal-external layout, either $N_i$ or $N_j$ must be reinserted in set $\mathcal{N}_a$. But this makes set $\mathcal{N}_a$ nonplanar. Therefore the channel does not adopt internal-external layout. The time complexity of Algorithm 5 is clearly $O(m)$.

4. CONCLUDING REMARKS

In this paper, we considered channels with interchangeable terminals and we considered various optimization problems. In the density minimization problem, we focused on the special case where there is only one cell per channel side. For that special instance, an approximation algorithm has been recently presented [5]. Our algorithm is optimal and asymptotically faster than the algorithm in [5]. The authors in [5] also proposed a variation of the above channel routing problem where one of the two cells has fixed terminals. In this paper, we showed that the latter problem is NP-complete. Many of the presented algorithms can be modified to minimize similarly defined objective functions for switchbox routing. Then we considered three more objective functions. We gave linear time algorithms for the via minimizing problem and the problem of determining whether the input instance adopts external-internal layout. Finally, we presented a polynomial time algorithm for the problem of finding a maximum planar subset of nets.

In the literature, many algorithms and heuristics have been proposed for a similar but different channel routing problem with movable terminals. Now the terminals are movable but their relative order remains unchanged in their cells. The algorithms and heuristics for this problem do not apply in our channel routing problem with interchangeable terminals. Below we briefly review work for this problem. Widmayer and C.K. Wong [28], Gopal, Coppersmith and C.K. Wong [9], Deogun and Bhattacharya [7], and Schlag, Woo and C.K. Wong [24] present algorithms for various combinatorial optimization problems for the latter channel routing problem. In [7, 16] optimal algorithms are given for switchbox routing with movable terminals. Cai and D.F. Wong [3] considered a more general instance of the channel problem in which position and order constraints exist for the terminals. Cai and D.F. Wong [3] considered a more general problem [3]. However, if the relative order of the terminals is fixed, Cai and Wong [3] present polynomial time algorithms that minimize the density. Similar channel routing problems have been studied in [1, 13].

5. ACKNOWLEDGEMENTS

I acknowledge useful discussions with Dimitri Kagaris and Ron Greenberg as well as very constructive comments and observations from the anonymous referees.

References


Biographies

SPYROS TRAGOUODAS received his Diploma degree in Computer Engineering from the University of Patras, Greece in July 1986 and his M.S. degree in Computer Science from the University of Texas at Dallas in August 1988. He received his Ph.D. degree in Computer Science from the University of Texas at Dallas in August 1991. He joined the faculty of Southern Illinois University at Carbondale in August 1991, where he is currently an Assistant Professor of Computer Science.

His research interets include Computer Aided Design for VLSI layout and Built-In Self Testing, algorithms for combinatorial optimization problems. Dr. Tragoudas is a member of IEEE Computer and CAS societies and SIGDA. In June 1990 he received a SIGDA-DATC Design Automation graduate scholarship.
Submit your manuscripts at http://www.hindawi.com