EDGECOINT PATHS IN A GRID BOUNDED BY TWO NESTED RECTANGLES

Hitoshi SUZUKI, Akira ISHIGURO* and Takao NISHIZEKI
Department of Electrical Communications, Faculty of Engineering, Tohoku University,
Sendai 980, Japan

Received 30 May 1989

This paper presents an algorithm for finding edge-disjoint paths in a given plane grid bounded by two nested rectangles. A pair of vertices on the boundary of the same rectangle are designated as terminals for each of the paths. The number of terminals lying on each boundary vertex is determined by the degree of the boundary vertex. Every vertex of degree 2 has either 0 or 2 terminals lying on it, every vertex of degree 3 has exactly one terminal, and every vertex of degree 4 has no terminal. If there are \( n \) vertices in the grid and \( b_1 \) vertices on the outer rectangle, then the algorithm decides in \( O(b_1) \) time whether there exist edge-disjoint paths, and actually finds the paths in \( O(n) \) time if there exist.

1. Introduction

In this paper we present an efficient algorithm for finding edge-disjoint paths, each connecting a terminal pair, in a given routing region. The region is modeled by a plane grid bounded by two nested rectangles, inner and outer, as shown in Fig. 1. Any two vertices on the boundary of the same rectangle can be designated as a terminal pair if the following constraint holds for every vertex \( v \): The number \( \delta(v) \) of terminals on a vertex \( v \) satisfies

\[
\delta(v) = \begin{cases} 
0 \text{ or } 2, & \text{if } \text{deg}(v) = 2, \\
1, & \text{if } \text{deg}(v) = 3, \\
0, & \text{if } \text{deg}(v) = 4,
\end{cases}
\]

where \( \text{deg}(v) \) is the degree of \( v \), i.e., the number of edges incident with \( v \). The grid depicted in Fig. 1 satisfies the constraint (1) above, and has edge-disjoint paths drawn in thick lines. Throughout the paper we denote by \( n \) the number of vertices in a grid, by \( b_1 \) the number of vertices on the outer rectangle and by \( k \) the number of terminal pairs. Our algorithm decides in \( O(b_1) \) time whether there exist \( k \) edge-disjoint paths in a given grid, and actually finds the paths in \( O(n) \) time if there exist.

The grid depicted in Fig. 2 does not satisfy the constraint (1) since there are several

* Current address: Nippon Denso Co., Ltd., Kariya 448, Japan.
vertices $v$ such that $\deg(v) = 2$ and $\delta(v) = 1$ or $\deg(v) = 3$ and $\delta(v) = 0$. However the following inequality holds for every vertex $v$:

$$\delta(v) \leq \begin{cases} 
2, & \text{if } \deg(v) = 2, \\
1, & \text{if } \deg(v) = 3, \\
0, & \text{if } \deg(v) = 4.
\end{cases}$$

(2)

We give a strong sufficient condition for the existence of edge-disjoint paths in a grid satisfying (2).
There are several known algorithms of polynomial time complexity which find edge-disjoint paths in various classes of grids or planar graphs having terminals only on the outer boundary [2, 4, 6, 8–10]. The algorithms in [4, 9] run for rectangular grids in $O(n^2)$ and $O(k \log k)$ time, respectively. The algorithm in [10] runs for "convex" grids including rectangular and L-shaped grids in $O(n)$ time (see also [7]). The algorithms in [2, 6] run for general grids in $O(n^{3/2})$ and $O(n \log^2 n)$ time, respectively. The algorithm in [8] runs for planar graphs in $O(n^2)$ time. However none of these algorithms can solve our problem because in our problem terminals may lie on both inner and outer rectangular boundaries. On the other hand the algorithm in [13, 15] for finding multicommodity flows in planar graphs can be applied to our problem, but it requires $O(n^2)$ time. Our algorithm is entirely different from the flow algorithm, and its complexity compares favorably with the flow algorithm.

The edge-disjoint paths found by our algorithm may contain a "knock-knee", that is, two paths may share turning points. Therefore the edge-disjoint paths cannot always be wired in two conductive layers in the knock-knee mode, but can be wired in at most four conductive layers [3]. Thus the algorithm is expected to be useful for the multi-layer routing problem of VLSI circuits.

2. Preliminaries

In this section we define relevant terms and review known results. Let $G = (V,E)$ be a finite undirected simple graph of vertex set $V$ and edge set $E$. Let $P$ be a set of terminal pairs $(t, t')$ where $t, t' \in V$, and let $k = |P|$. The pair $N = (G, P)$ is called a network. A path connecting vertices $u$ and $o$ is called a $U-U$ path. A set of $k$ pairwise edge-disjoint $t-t'$ paths in $G$ for all $(t, t') \in P$ are called edge-disjoint paths of network $N$.

For $X, Y \subseteq V$, define

$$C(X; Y) = \{(u, u') \in E : u \in X, u' \in Y\},$$
$$D(X; Y) = \{(t, t') \in P : t \in X, t' \in Y\},$$
$$c(X; Y) = |C(X; Y)|,$$
$$d(X; Y) = |D(X; Y)|.$$

For $X \subseteq V$, define

$$C(X) = C(X; V - X), \quad D(X) = D(X; V - X),$$
$$c(X) = c(X; V - X), \quad d(X) = d(X; V - X), \quad m(X) = c(X) - d(X).$$

$C(X)$ defined above is called a cut, and especially called a cutset if the graph $G - C(X)$ obtained from $G$ by deleting the edges in $C(X)$ has exactly one more connected component than $G$. On the other hand $c(X)$, $d(X)$ and $m(X)$ are called the
capacity, demand and margin of cut \( C(X) \), respectively. We say that a network \( N \) satisfies the cut condition if \( m(X) \geq 0 \) for every cut \( C(X) \). A cut \( C(X) \) is saturated if \( m(X) = 0 \). Clearly the cut condition is necessary for the existence of edge-disjoint paths in a network \( N \), but not always sufficient. The following lemma has been known.

**Lemma 2.1** [12]. A network \( N = (G, P) \) satisfies the cut condition if and only if \( m(X) \geq 0 \) for every cut \( C(X) \).

For a subgraph \( H \) of \( G \), \( V(H) \) denotes the set of vertices of \( H \), and \( E(H) \) the set of edges of \( H \). \( N = (G, P) \) is called a planar network if

(a) \( G \) is a planar graph with two specified face boundaries \( B_1 \) and \( B_2 \); and

(b) every pair of terminals lie on the same boundary \( B_1 \) or \( B_2 \), that is, \( P \) is partitioned into \( P_1 \) and \( P_2 \) so that

\[
(t, t') \in P_1 \Rightarrow t, t' \in V(B_1);
\]

\[
(t, t') \in P_2 \Rightarrow t, t' \in V(B_2).
\]

Denote by \( n, b_1 \) and \( b_2 \) the number of vertices of \( G \), \( B_1 \) and \( B_2 \), respectively. One may assume without loss of generality that \( B_1 \) is the outer face boundary of \( G \). The extended degree \( \deg^*(u) \) of a vertex \( u \) is defined as follows: \( \deg^*(u) = \deg(u) + \delta(u) \), where \( \deg(u) \) is the degree of \( u \) in \( G \) and \( \delta(u) \) is the number of terminals on \( u \). A network \( N \) is even if \( \deg^*(u) \) is even for every vertex \( u \in V \). Okamura has obtained the following elegant theorem for even planar networks [11].

**Theorem 2.2** [11]. An even planar network has edge-disjoint paths if and only if the network satisfies the cut condition.

Let \( G^+ \) be the plane grid, i.e., the infinite graph whose vertex set consists of all points of the plane with integer coordinates and in which two vertices are connected if and only if the Euclidean distance between them is equal to 1. A planar network \( N = (G, P) \) is called a grid network if

(a) \( G = (V, E) \) is a finite subgraph of \( G^+ \) bounded by two nested rectangles, the outer called \( B_1 \) and the inner called \( B_2 \);

(b) \( \deg(v) - 4 \) for every \( v \in V - V(B_1) \cup V(B_2) \); and

(c) the constraint (2) in the Introduction holds.

Our main objective is to find edge-disjoint paths in a given even grid network \( N \). Note that any even grid network \( N \) satisfies the constraint (1), and that if \( \delta(u) = 0 \) for a vertex \( u \in V(B_1) \cup V(B_2) \), then either (i) \( u \) is one of the corners of rectangles \( B_1 \) and \( B_2 \) or (ii) \( u \in V(B_1) \cap V(B_2) \) and \( \deg(u) = 2 \).

One may assume \( b_1, b_2 = O(k) \) for an even grid network \( N \) as follows. Since \( b_1 \geq b_2 \), we shall show that \( b_1 = O(k) \). Consider all the vertical or horizontal lines \( L \)
that pass through two vertices on $B_1$, other than the four corners of rectangle $B_1$. If at least one terminal lies on each of these lines, then clearly $b_1 = O(k)$. Thus there is a line $L$ on which no terminal lies. Then the two vertices $u$ and $v$ on the intersections of $B_1$ and $L$ have degree two, and must lie on $B_2$ as shown in Fig. 3(a). Delete $u$ and $v$ and replace two edges incident with $u$ or $v$ by a single edge, as shown in Fig. 3(b). Clearly the resulting network $N'$ is also an even grid network, and has edge-disjoint paths if and only if the original network $N$ has. Repeatedly applying the procedure above, we can eventually obtain an even grid network for which at least one terminal lies on each of these lines $L$. Clearly $b_1 = O(k)$ in the network. The reduction above can be done in $O(b_1)$ time. In fact it can be done in $O(k + \text{SORT}) = O(\min\{b_1, k \log k\})$ time where $\text{SORT} = O(\min\{b_1, k \log k\})$ denotes the time necessary to sort the terminals on $B_1$ or $B_2$, that is, order them clockwise around $B_1$ or $B_2$. Refer to [1] for standard sorting algorithms.

3. Existence of paths in even grids

According to Theorem 2.2 and Lemma 2.1, one can check the existence of edge-disjoint paths in an even grid network by verifying whether $m(X) \geq 0$ for every cutset $C(X)$. This can be done in $O(n^2)$ time for an even planar network by the multicommodity flow algorithm in [13, 15]. In this section we first give a simple form of Theorem 2.2 for even grid networks. Then, using this form, we give an $O(b_1)$ algorithm to check the existence of paths in even grid networks.

3.1. Simple form of Theorem 2.2

The cutsets $C(X)$ of an even grid network $N=(G,P)$ are classified into four types, depending on whether
Note that $|C(X) \cap E(B_1)|$ and $|C(X) \cap E(B_2)|$ are either 0 or 2 for every cutset $C(X)$.

Since either $t, t' \in V(B_1)$ or $t, t' \in V(B_2)$ for every pair $(t, t') \in P$, a cutset $C(X)$ of type (0) satisfies $d(X) = 0$ and hence $m(X) \geq 0$. On the other hand by the constraint (1) a cutset $C(X)$ of type (1) satisfies $c(X) \geq d(X) + 2$ and hence $m(X) \geq 2$. These observations yield the following lemma.

**Lemma 3.1.** An even grid network $N$ satisfies the cut condition if and only if $m(X) \geq 0$ for all cutsets $C(X)$ of types (2) and (3).

Thus it suffices to compute the margins of cutsets of types (2) and (3). We next show that in fact it suffices to compute the margins of “rectilinear” and “box” cuts of these types. As illustrated in Fig. 4, a **rectilinear cut** is a cutset consisting of edges crossing a horizontal or vertical straight line that does not cross $B_2$. We denote the four sides of rectangle $B_1$ by $B_1^1$, $B_1^2$, $B_1^3$ and $B_1^4$ in the clockwise order. Similarly we denote those of rectangle $B_2$ by $B_2^1$, $B_2^2$, $B_2^3$ and $B_2^4$. Assume that $B_1^1$ and $B_1^2$ are the top sides of $B_1$ and $B_2$, respectively. Let $L$ be a vertical or horizontal line segment intersecting with $B_1$ and $B_2$ at exactly two edges $e \in B_1^2$ and $e' \in B_1^1, 1 \leq i \leq 4$. Define **semicut** $C_i(e)$ to be the set of edges including $e$ and $e'$ which are intersecting with $L$ and lie between $e$ and $e'$. A semicut is illustrated in Fig. 4. A union of two
distinct semicuts $C_i(e_1) \cup C_i(e_2)$ is called a box cut, and is denoted by $C_i(e_1, e_2)$ where $e_1, e_2 \in E(B_i)$.

For two edges $e$ and $e'$ on $B_i^j$, $1 \leq i \leq 4$, let $l_i^j(e, e')$ denote the number of vertices on the path on $B_i^j$ from an end of $e$ to an end of $e'$ passing through neither $e$ nor $e'$. In particular, $l_i^j(e, e) = 0$ for each edge $e \in B_i$. We now have the following lemma.

**Lemma 3.2.** In an even grid network $N$ all cutsets $C(X)$ of type (2) satisfy $m(X) \geq 0$ if and only if all rectilinear cuts $C(Y)$ satisfy $m(Y) \geq 0$.

**Proof.** We shall show that for any cutset $C(X)$ of type (2) either (i) $m(X) \geq 0$ or (ii) there is a rectilinear cut $C(Y)$ such that $m(X) > m(Y)$. Let $\{e, e'\} = C(X) \cap E(B_i)$. If both $e$ and $e'$ lie on the same side $B_i^j$, $1 \leq i \leq 4$, then clearly $c(X) \geq d(X) + 2$ and hence $m(X) > 0$. Similarly if $e$ and $e'$ lie on two adjacent sides $B_i^j$ and $B_i^{j+1}$, $1 \leq i \leq 4$, respectively, then $c(X) \geq d(X)$ and hence $m(X) \geq 0$ where $B_i^j = B_i$. Thus we may assume that $e \in E(B_i^j)$ and $e' \in E(B_i^{j+2})$, $i = 1$ or $2$. Let $C(Y)$ be any rectilinear cut such that $C(Y) \cap C(X) \neq \emptyset$ and $C(Y) \cap E(B_i^j) \neq \emptyset$. Let $\{a\} = C(Y) \cap E(B_i^j)$ and $\{a'\} = C(Y) \cap E(B_i^{j+2})$. Then it is straightforward to verify that

$$c(X) \geq c(Y) + l_i^j(e, a) + l_i^{j+2}(e', a')$$

and

$$d(X) \leq d(Y) + l_i^j(e, a) + l_i^{j+2}(e', a').$$

Therefore $m(X) \geq m(Y)$. \qed

For two edges $e$ and $e'$ on $B_2$, let $l_2^j(e, e')$ denote the number of vertices on the shortest path on $B_2$ from an end of $e$ to an end of $e'$ passing through neither $e$ nor $e'$. For two distinct edges $e$ and $e'$ on $B_2$, define $l_2(e, e') = (l_2^j(e, e')$ minus the number of corners of rectangle $B_2$ through which the shortest path passes), while define $l_2(e, e) = 0$ for each edge $e \in E(B_2)$. The following lemma holds.

**Lemma 3.3.** In an even grid network $N$ all cutsets $C(X)$ of type (3) satisfy $m(X) \geq 0$ if and only if all box cuts $C(Y)$ satisfy $m(Y) \geq 0$.

**Proof.** We shall show that for any cutset $C(X)$ of type (3) either (i) $m(X) \geq 0$ or (ii) there is a box cut $C(Y)$ such that $m(X) > m(Y)$. Let $\{e_1, e'_1\} = C(X) \cap E(B_1)$, $\{e_2, e'_2\} = C(X) \cap E(B_2)$, $e_1 \in E(B_1^j)$ and $e'_1 \in E(B_1^{j+2})$, $1 \leq i, j \leq 4$, as depicted in Fig. 5. One may assume without loss of generality that the clockwise right ends of edges $e_1$ and $e_2$ belong to $X$. There are semicuts $C_i(a_1)$ and $C_i(a_2')$ crossing $B_1^j$ and $B_2^j$, respectively, where $a_2 \in E(B_1^j)$ and $a'_2 \in E(B_2^j)$. Choose $C_i(a_2)$ and $C_i(a_2')$ such that $l_i^j(e_1, a_1)$ and $l_i^j(e'_1, a'_1)$ are minimum where $\{a_1\} = E(B_1^j) \cap C_i(a_1)$ and $\{a'_1\} = E(B_1^{j+2}) \cap C_i(a_2')$. 


First consider the case $a_2 \neq a'_2$, as illustrated in Fig. 5(a). Then $C(Y) = C_1(a_2) \cup C_2(a'_2)$ is a box cut. It is straightforward to verify that
\[c(X) \geq c(Y) + l_1(e_1, a_1) + l_2(e_2, a_2) + l_1(e'_1, a'_1) + l_2(e'_2, a'_2).\]
Furthermore
\[ d(X) \leq d(Y) + l_1'(e_1, a_1) + l_2(e_2, a_2) + l_1(e_1', a_1') + l_2(e_2', a_2'). \]

This inequality follows from the constraint (1) and the facts (a) and (b) given below:

(a) If a corner \( v \) of the inner rectangle \( B_2 \) is not on \( B_1 \), then there is no terminal on \( v \), so \( v \) has no contribution to \( d(X) \) or \( d(Y) \);

(b) if the path on \( B_2 \) attaining \( l_2(e_2, a_2) \) or \( l_2(e_2', a_2') \) passes through a corner \( v \) of \( B_2 \), then there is a terminal \( t \) on \( v \), but the path attaining \( l_1'(e_1, a_1) \) or \( l_1'(e_1', a_1') \) also passes through \( v \), so the contribution of \( (t, t') \) to \( d(X) \) is accounted for by \( l_1'(e_1, a_1) \) or \( l_1'(e_1', a_1') \) in the right-hand side of the inequality above.

Therefore \( m(X) \geq m(Y) \).

Next consider the case \( a_2 = a_2' \), as illustrated in Fig. 5(b). Then \( C_4(a_2) \cup C_4(a_2') \) is not a box cut, but one can observe
\[ c(X) = |C_4(a_2)| + |C_4(a_2')| + l_1'(e_1, a_1) + l_2(e_2, a_2) + l_1'(e_1', a_1') + l_2(e_2', a_2'); \]
\[ d(X) \leq l_1'(e_1, a_1) + l_2(e_2, a_2) + l_1'(e_1', a_1') + l_2(e_2', a_2'). \]

Therefore \( m(X) \geq 0 \).

Thus we have the following simple form of Theorem 2.2.

**Theorem 3.4.** An even grid network \( N \) has edge-disjoint paths if and only if all rectilinear and box cuts of \( N \) have nonnegative margins.

**3.2. Checking the existence**

In this subsection we show how to efficiently compute the margins of rectilinear and box cuts. One can compute the capacity \( c(X) \) of a rectilinear or box cut \( C(X) \) in a constant time if the two edges in \( C(X) \cap E(B_1) \) are given. Moreover one can easily compute the margins of all rectilinear cuts in \( O(k) \) time. Remember that \( b_1, b_2 = O(k) \). Thus we shall concentrate on box cuts. Let the vertices and edges \( u_0, v_0, u_1, v_1, \ldots, u_{b_2-1}, v_{b_2-1} \) of \( B_2 \) appear in this order clockwise on \( B_2 \). The capacity, demand and margin of a box cut \( C_b(e_i, e_j), e_i, e_j \in B_2 \), are denoted by \( c_b(e_i, e_j) \), \( d_b(e_i, e_j) \) and \( m_b(e_i, e_j) \), respectively. For convenience define \( d_b(e_i, e_i) = 0 \) and \( c_b(e_i, e_i) = 2|C_b(e_i)| \), and hence \( m_b(e_i, e_i) = 2|C_b(e_i)| \). Clearly one can compute in \( O(k) \) time margins \( m_b(e_i, e_j) \) for a fixed edge \( e_i \) and all edges \( e_j, 0 \leq j \leq b_2 - 1 \). Therefore, by repeating the computation for each edge \( e_i \), one can compute in \( O(k^2) \) time the margins of all box cuts. Thus it is rather straightforward to show that the existence of paths can be checked in \( O(k^2) \) time for an even grid network.

We next improve the complexity \( O(k^2) \) above to \( O(k \log k) \). Let \( M = \text{MIN}\{m_b(e_i, e_j) : e_i, e_j \in E(B_2)\} \), and let \( M^{(r)} = \text{MIN}\{m_b(e_i, e_j) : e_i \in E(B_2^r), e_j \in E(B_2)\} \) for each side \( B_2^r \) of the rectangle \( B_2 \), \( 1 \leq r \leq 4 \). Then \( M = \text{MIN}\{M^{(1)}, M^{(2)}, M^{(3)}, M^{(4)}\} \). Thus we
shall show that each $M^{(r)}$ can be computed in $O(k \log k)$ time. In order to compute $M^{(r)}$, we first compute values $m_b(e_i, e_j)$ for a fixed edge $e_i \in E(B'_1)$ and all edges $e_j \in E(B_2)$, and then iteratively update these values for the clockwise next edge $e_{i+1}$ on the same side $B'_2$. By definition

$$m_b(e_{i+1}, e_j) = m_b(e_i, e_j) + \{ c_b(e_{i+1}, e_j) - c_b(e_i, e_j) \} + \{ d_b(e_i, e_j) - d_b(e_{i+1}, e_j) \}. $$

Since $e_i$ and $e_{i+1}$ are on the same side $B'_2$, $c_b(e_{i+1}, e_j) - c_b(e_i, e_j) = 0$. On the other hand $d_b(e_i, e_j) - d_b(e_{i+1}, e_j)$ takes one of three fixed values depending on the location of $e_j$. Let $v_{i+1}'$ be the vertex on $B'_1$ corresponding to $v_{i+1}$ on $B'_2$. Then $B_2$ is partitioned into three intervals according to the location of the terminals of $D\{u_{i+1}, v_{i+1}'\}$. For example, in Fig. 6 $B_2$ is partitioned into three intervals A, B and C according to mates $t'_1$ and $t'_2$ of terminals $t_1$ on $v_{i+1}'$ and $t_2$ on $v_{i+1}$ respectively, and

$$d_b(e_i, e_j) - d_b(e_{i+1}, e_j) = \begin{cases} 2, & \text{if } e_j \text{ is on interval A}, \\ 0, & \text{if } e_j \text{ is on interval B}, \\ -2, & \text{if } e_j \text{ is on interval C}. \end{cases}$$  

We need an appropriate data structure in order to efficiently update $m_b$ and compute $\text{MIN}\{ m_b(e_i, e_j); e_j \in E(B_2) \}$. The simplest one is a height-balanced binary tree $T$ whose leaves correspond to $e_j, 0 \leq j \leq b_2 - 1$. Each node $v$ of $T$ has two keys $\text{key}_1(v)$ and $\text{key}_2(v)$. These keys are first initialized as follows:

$$\text{key}_1(v) = \begin{cases} m_b(e_i, e_j), & \text{if } v \text{ is a leaf and corresponds to } e_j, \\ \text{MIN}\{ m_b(e_i, e_j); e_j \text{ corresponds to a leaf descendant of } v \} & \text{in } T, \\ 0, & \text{if } v \text{ is an internal node of } T. \end{cases}$$

$$\text{key}_2(v) = 0, \text{ for every node } v \text{ of } T.$$

Fig. 6. Difference of demands due to pairs $(t_1, t'_1)$ and $(t_2, t'_2)$. 
Whenever $i$ is increased, the keys are updated as follows. If all the leaf descendants of node $v$ in $T$ correspond to edges in interval $A$ and $v$'s father does not satisfy this property, then $k_2(v)$ is increased by 2 (see equation (3)). Furthermore, for each node $u$ on the path from $v$ to the root in $T$, $k_1(u)$ is updated so that the following equation holds:

$$\text{MIN}\{m_b(e_i,e_j): e_j \text{ corresponds to a leaf descendant of node } u\} = k_1(u) + \sum \{k_2(w): w \text{ is a node on the path from } u \text{ to the root}\}.$$ 

Similar updating must be done for intervals $B$ and $C$ as well. Since the length of a path from the root to a leaf in $T$ is $O(\log k)$, one can update the keys in $T$ and compute $\text{MIN}\{m_b(e_i,e_j): e_j \in E(B_2)\}$ (=$k_1(\text{root})+k_2(\text{root})$) for a fixed edge $e_i$ in $O(\log k)$ time. Since $|E(B'_2)|=O(k)$, $M^r = \text{MIN}\{m_b(e_i,e_j): e_i \in E(B'_2), e_j \in E(B_2)\}$ can be computed in $O(k \log k)$ time. Thus one can compute $M$ and hence check the existence of paths in $O(k \log k)$ time.

In place of a binary tree one can use a more sophisticated data structure such as a "variable-priority queue" [14]. The queue supports the FIND-MIN operation together with the key updating described above, and executes $k$ operations in $O(k)$ time. The variable-priority queue uses the linear time algorithm for a special case of disjoint set union [5]. Thus one can check the existence of paths in $O(k \log k)$ time or indeed $O(\text{MIN}\{b_1,k \log k\})$ time. Remember that $\text{SORT} = \text{MIN}\{b_1,k \log k\}$.

4. Finding paths in even grids

In this section we present an efficient algorithm for finding edge-disjoint paths in a given even grid network satisfying the cut condition. In Subsection 4.1 we present two procedures SEARCH and CUTOFF. For convenience we describe these procedures so that they work not only for even grid networks but also for even planar networks. In Subsection 4.2 we present an algorithm for even grid networks using these procedures. Finally in Subsection 4.3 we show that the algorithm runs in $O(n)$ time.

4.1. SEARCH and CUTOFF

Trivially the following lemma holds.

**Lemma 4.1.** If a network $N$ has edge-disjoint paths and has a saturated cutset $C(X)$, then all edges in $C(X)$ are used by the paths connecting pairs in $D(X)$ and each of these paths uses exactly one of the edges in $C(X)$.

In what follows, we assume that $N=(G,P)$ is an even planar network satisfying the cut condition. For an edge $e=(v,v') \in E(B_i)$, $i=1$ or 2, procedure CUTOFF($N$, $e,B_i$) described later executes one of the following operations (a), (b) and (c):
(a) if edge $e$ is redundant in $N$, that is, $N$ has edge-disjoint paths not using $e$, then CUTOFF deletes $e$ from $G$;
(b) if $N$ has edge-disjoint paths including a $t-t'$ path using $e$ for a pair $(t, t') \in P_i$, then CUTOFF deletes and "reserves" $e$ for the $t-t'$ path;
(c) otherwise, CUTOFF reports that $e$ must be used by a path for a terminal pair in $P-P_i$.

By the following Lemmas 4.2 and 4.3, one can know which operation (a), (b) or (c) must be executed for an edge $e \in B_i$, $i = 1$ or 2 and which pair must be chosen as $(t, t')$ if operation (b) is executed. From now on, the margin of a cut $C(X)$ in a new network $N'$ is denoted by $m'(X)$.

**Lemma 4.2.** An edge $e=(u, v')$ on $B_i$ is redundant in $N$ if and only if $N$ has no saturated cutset $C(X)$ such that $e \in C(X)$.

**Proof.** If $N$ has a saturated cutset $C(X)$ such that $e \in C(X)$, then, by Lemma 4.1, $e$ must be used by a path connecting a pair in $D(X)$ and hence $e$ is not redundant. Suppose that $N$ has no saturated cutset $C(X)$ such that $e \in C(X)$. Construct a new network $N'$ from $N$ as follows:

(i) delete $e$ from $G$; and
(ii) add a dummy terminal pair $(u, u')$ to $P$.

That is, $N'=(G-e, P+(u, u'))$. Since $N$ is an even planar network, so is $N'$. Therefore it suffices to prove that $N'$ satisfies the cut condition. Because then, by Theorem 2.2, $N'$ has edge-disjoint paths including a $u-v'$ path for the dummy pair $(u, v')$ and hence $N$ has edge-disjoint paths not using $e$, that is, $e$ is redundant in $N$. Since $N$ is even, one can show with an easy parity argument that the margin of any cut of $N$ is an even integer. Since $N$ satisfies the cut condition, the margin of any cut of $N$ is a nonnegative integer. Let $C(X)$ be any cutset of $N$ such that $e \in C(X)$; then $m(X) \geq 2$ since $C(X)$ is not saturated. Therefore the margin $m'(X)$ of the cutset $C(X)$ of $N'$ satisfies $m'(X) = \{c(X) - 1\} - \{d(X) + 1\} = m(X) - 2 \geq 0$. Clearly any other cutset $C(X)$ of $N'$ satisfies $m'(X) = m(X) \geq 0$. Thus network $N'$ satisfies the cut condition, as desired.

Thus when (a) is executed, a new network $N'$ is constructed from $N$ not only by deleting $e$ from $G$ but also by adding a dummy pair $(u, v')$ to $P$. A $v-v'$ path for the dummy pair will be found in $N'$ but simply discarded later.

**Lemma 4.3.** Let $e=(u, v')$ be a nonredundant edge on $B_i$, and let $(t, t')$ be a terminal pair in $P_i$. Suppose that $t, v, v'$ and $t'$ appear in this order clockwise on $B_i$. $N$ has edge-disjoint paths including a $t-t'$ path using $e$ if and only if $N$ has no saturated cutset $C(Y)$ such that $t, t' \notin Y$ and $\{v, v'\} \cap Y \neq \emptyset$. 


Proof. First we prove the necessity. Suppose that there is a saturated cutset \( C(Y) \) such that \( t, t' \notin Y \) and \( \{u, u'\} \cap Y \neq \emptyset \). Either \(|\{u, u'\} \cap Y| = 1 \) or \( 2 \). Lemma 4.1 implies that

(a) if \(|\{u, u'\} \cap Y| = 2\), then edge \( e = (u, u') \) must be used by a path for a pair in \( D(Y; V) \); and

(b) if \(|\{u, u'\} \cap Y| = 1\), then edge \( e \) must be used by a path for a pair in \( D(Y) \).

Thus the \( t-t' \) path cannot use \( e \) since \((t, t') \notin D(Y) \cup D(Y; V)\).

Next we prove the sufficiency. Suppose that there is no saturated cutset \( C(Y) \) such that \( t, t' \notin Y \) and \( \{u, u'\} \cap Y \neq \emptyset \). Since \( N \) is even, \( m(Y) \geq 2 \) if \( t, t' \notin Y \) and \( \{u, u'\} \cap Y \neq \emptyset \). Let \( N' \) be the network obtained from \( N \) by removing edge \( e \) and replacing terminal pair \((t, t')\) by two terminal pairs \((t, u)\) and \((u', t')\), that is, \( N' = (G - e, P - (t, t') + (t, u) + (u', t')) \). It is sufficient to show that the even planar network \( N' \) satisfies the cut condition. Because then \( N' \) has edge-disjoint paths, and the desired edge-disjoint paths of \( N \) can be constructed from them by replacing the \( t-u \) and \( u'-t' \) paths in \( N' \) with the \( t-t' \) path obtained by joining the two paths via \( e \). Let \( C(Y) \) be a cutset of \( N' \). One may assume that \( \{u, u'\} \cap Y \neq \emptyset \): otherwise consider the cutset \( C(Y) \) of \( N' \). Then

\[
m'(y) = \begin{cases} 
    m(y) - 2, & \text{if } t, t' \notin Y, \\
    m(y), & \text{otherwise}.
\end{cases}
\]

Therefore \( m'(Y) \geq 0 \). □

When (b) is executed in CUTOFF, a new network \( N' = (G - e, P - (t, t') + (t, u) + (u', t')) \) is constructed. After edge-disjoint paths of \( N' \) are found, the \( t-t' \) path will be constructed by concatenating the \( t-u \) and \( u'-t' \) paths via the “reserved” edge \( e \).

When there is a saturated cutset \( C(X) \) such that \( e = (u, u') \in E(B_i) \cap C(X) \), edge \( e \) must be used by a path for a pair in \( D(X) \). Therefore, by verifying for each pair \((t, t') \in D(X) \cap P_i\) whether \( m(Y) > 0 \) for all cutsets \( C(Y) \) such that \( t, t' \notin Y \) and \( \{u, u'\} \cap Y \neq \emptyset \), one can determine whether a path for \( P_i \) uses edge \( e \) or that there does not exist such a path. The function \( \text{SEARCH}(N, e, X) \) given below returns a pair in \( D(X) \cap P_i \) if a path for the pair can use edge \( e \), or returns \( \text{nil} \) if none of the paths for \( P_i \) can use edge \( e \). Let \( e_0 = e, e_1, \ldots, e_{b-1} \) be the clockwise sequence of edges on \( B_i \) as shown in Fig. 7. Let \( e_j = (v_j, v_{j+1}), 0 \leq j \leq b-1 \), \( e_b = e_0 \) and \( v_b = v_0 \). Define

\[
m(q, r) = \min\{m(Y): C(Y) \cap E(B_i) = \{e_q, e_r\}, \quad 0 \leq q, r \leq b-1, \}
\]

\[
I(q) = \{v_j: 1 \leq j \leq q\}, \quad 1 \leq q \leq b-1, \\
\text{index}(v_q) = q, \quad 0 \leq q \leq b-1.
\]

Furthermore define for a pair \((t, t') \in P_i\) and \( 0 \leq q \leq \text{index}(t') - 1 \)

\[
m_q(t) = \min\{m(q, r): \text{index}(t) \leq r \leq b\}.
\]
Thus a $t-t'$ path can use edge $e$ if and only if $m_i(q) > 0$ for every $q$, $0 \leq q \leq \text{index}(t') - 1$.

**function** SEARCH($N, e, X$);
**begin**
\{ $e \in E(B_i) \cap C(X)$, $m(X) = 0$ and $e = e_0$ \}
$Q := D(X) \cup P_i$; \{ $Q \subset P_i$ is the set of pairs which may be connected by a path using $e$ \}
**while** $Q \neq \emptyset$ **do**
**begin**
let $(t, t')$ be the pair in $Q$ such that $t$ or $t'$, say $t$, is the terminal that appears first on the boundary $B_i$ counterclockwise starting from $v_0$;
$q := \text{index}(t') - 1$;
**while** $m_i(q) > 0$ and $q \geq 0$ **do** $q := q - 1$;
**if** $q < 0$ **then** \{ $m_i(q) > 0$ for every $q$, $0 \leq q \leq \text{index}(t') - 1$ \}
\{ $m_i(q) > 0$. See Comment given below \}
$Q := Q \cap D(U(q))$;
**end**;
**return**(nil) \{ none of the paths for $P_i$ can use $e$ \}
**end**;

**Comment.** Since there is a saturated cutset $C(Y)$ such that $C(Y) \cap E(B_i) = \{ e_q, e_r \}$, index($t$) $\leq r \leq b$, by Lemma 4.3 the $t-t'$ path cannot use edge $e$. Because of the selection of $(t, t')$, every path for $Q - D(l(q))$ is blocked by the same saturated cutset $C(Y)$. Therefore a path which may use $e$ has a terminal on $v_1, v_2, \ldots$, or $v_q$ (see Fig. 7).

Let $e = (v_i, v') \in E(B_i)$, $i = 1$ or 2, and assume that $v$, $e$ and $v'$ appear on the boundary $B_i$ clockwise in this order. Define
\[ m(e) = \text{MIN}\{m(X); e \in C(X)\}. \]

We are now ready to present procedure \text{CUTOFF}(N, e, B_i),

\begin{verbatim}
procedure CUTOFF(N, e, B_i);
begin
  if \( m(e) > 0 \) then \{e is redundant\}
  begin
    \( G := G - e; \)
    \( P_i := P_i + (u, u') \)
    \{(u, u') is a dummy pair\}
  end
  else \{e is not redundant\}
  begin
    let \( C(X) \) be a saturated cutset with \( e \in C(X) \);
    \( p := \text{SEARCH}(N, e, X); \)
    \{if \( p = \text{nil} \) then e cannot be used by any path for \( P_i \}\)
    if \( p \neq \text{nil} \) then
    begin
      \{reserve edge e for the path for \( P_i \}\)
      let \( p = (t, t') \), and assume w.l.o.g. that \( t, u, v' \) and \( t' \) appear
      on the boundary \( B_i \) clockwise in this order;
      \( G := G - e; \)
      \( P_i := P_i - (t, t') + (t, v) + (v', t') \)
    end
  end;
end;
\end{verbatim}

One can find edge-disjoint paths in an even planar network by repeatedly applying \text{CUTOFF} for edges on \( B_1 \) or \( B_2 \) in a suitable order. However the algorithm is not as efficient as the known \( O(n^2) \) algorithm [13, 15].

4.2. Algorithm

In this subsection we first present an algorithm which finds edge-disjoint paths in an even grid network \( N \) using procedures \text{CUTOFF} and \text{SEARCH}, and then verify the correctness of the algorithm.

The main idea of our algorithm is to dig a “tunnel” joining \( B_1 \) and \( B_2 \) by applying \text{CUTOFF} to the edges in a semicut so that \( N \) is reduced to a new network having all terminals on a single boundary. This can be done by the following procedure \text{REDUCE}.
procedure REDUCE(N);
begin
choose an arbitrary edge $e_0$ on $B_2$;
$E' := C_2(e_0)$;
repeat {delete redundant edges in $E'$ or reserve edges in $E'$ for pairs in $P_1$}
let $e \in E' \cap B_1$; {although the outer boundary $B_1$ may be altered, we generously denote by $B_1$ the current outer boundary}
CUTOFF(N, e, $B_1$);
if $e$ was deleted then $E' := E' - e$;
until $e$ was not deleted or $E' = \emptyset$;
{reserve the remaining edges in $E'$ for pairs in $P_2$}
while $E' \neq \emptyset$ do
begin
let $e \in E' \cap E(B_2)$;
CUTOFF(N, e, $B_2$);
$E' := E' - e$
end
end;

For example, suppose that the semicut $C_2(e_0)$ is chosen in an even grid network $N$ depicted in Fig. 1, where $e_0$ is the leftmost edge on $B_2$. Then REDUCE(N) transforms $N$ into a new even network $N'$ depicted in Fig. 8 by applying CUTOFF to each of the three edges in the semicut. Let $e_{\text{top}}$, $e_{\text{mid}}$ and $e_{\text{bot}}$ be the three edges from top to bottom. None of them is redundant since the box cut $C(X) = C_2(e_0, e_6)$ is saturated, where $e_6$ is the rightmost edge on $B_2$. In the repeat statement

Fig. 8. Network obtained from an even grid network $N$ in Fig. 1 by applying REDUCE(N).
CUTOFF is executed as follows. First, CUTOFF(N,e\text{top},B_1) deletes e\text{top} and
replaces a pair (14, 14)\in P_1\cap D(X) with two new pairs (14', 14') and (14", 14").
Note that the two terminals of (14", 14") lie on the same vertex. Then, CUTOFF(N,
e\text{mid},B_1) deletes e\text{mid} and replaces (9, 9) with (9', 9') and (9", 9"). Next, CUTOFF(N,
e\text{bot},B_1) does not alter N at all since it is known that no path for P_1\cap D(X')
can use e\text{bot} where C(X') = C_b(e_0,e_2) is a saturated box cut, and hence e\text{bot} must be used
by a path for P_2. Now the repeat statement in REDUCE is completed. In the while
statement, the first and last execution of CUTOFF, i.e., CUTOFF(N,e\text{bot},B_2)
deletes e\text{bot} and replaces a pair (1, 1)\in P_2\cap D(X') with two pairs (1', 1') and (1", 1").
As shown later, REDUCE deletes all the edges in C_r(e_0) and hence reduces N to
a new even network N' having all terminals on the outer boundary. Therefore we
find edge-disjoint paths in N' by using any known algorithm given in [2, 6, 8], and
finally construct paths in N from the paths in N'.

We now verify the correctness of the algorithm above. It suffices to show that
if CUTOFF(N,e,B_r) called in REDUCE(N) does not delete edge e, then the
remaining edges e' in E' are all used by paths for P_2 and hence deleted by
CUTOFF(N,e',B_2) in the while statement. Noting the fact that N was originally a
grid network, one can easily prove the following lemma.

**Lemma 4.4.** Let N be a network appearing during the execution of REDUCE, and
let e\in E'\cap E(B_r). If N has a saturated cutset C(X) such that e\in C(X) and
C(X)\cap E(B_r) \neq \emptyset, then N has a saturated cutset C(X') such that E'\subset C(X').

**Lemma 4.5.** If CUTOFF(N,e,B_r) called in REDUCE(N) does not delete edge e,
then no edge in E' can be used by a path for P_1.

**Proof.** Since CUTOFF(N,e,B_r) does not delete e, there is a saturated cutset C(X)
such that $e \in C(X)$. $C(X)$ remains saturated during the execution of REDUCE since the margin of any cut does not increase. Since SEARCH called in CUTOFF returns $nil$, edge $e$ cannot be used by any path for $P_1$. Therefore $C(X) \cap E(B_2) \neq \emptyset$. By Lemma 4.4 we may assume that $E' \subset C(X)$, and hence by Lemma 4.1 edge $e$ must be used by a path for a pair $(t_2, t'_2) \in P_2 \cap D(X)$. Since $m(X) = 0$, every edge in $E'$ must be used by a path for $D(X)$. If an edge $e'_1 \in E'$, $e'_1 \neq e$, can be used by a path connecting a pair $(t_1, t'_1) \in P_1 \cap D(X)$, then these two paths intersect at two or more vertices, and can be replaced by a $t_1-t'_1$ path through $e$ and a $t_2-t'_2$ path through $e'_1$, as illustrated in Fig. 9. This is a contradiction. 

Thus we have the following lemma.

**Lemma 4.6.** Procedure REDUCE$(N)$ deletes all the edges in $C_3(e_0)$.

**Proof.** It is sufficient to prove that $e \in E(B_2)$ is deleted whenever CUTOFF$(N, e, B_2)$ is executed. If $e$ is not deleted, then $e$ cannot be used by any path for $P_2$. Moreover by Lemma 4.5 edge $e$ cannot be used by any path for $P_1$. However, since edge $e$ is not deleted, there is a saturated cutset $C(X)$ such that $e \in C(X)$, and hence edge $e$ must be used by a path for $P_1$ or $P_2$, a contradiction. 

### 4.3. Running time

We claim that REDUCE$(N)$ can be done in $O(kw)$ time where $w = |C_3(e_0)|$. The execution of procedure CUTOFF called in REDUCE is dominant in the computation time of REDUCE. Furthermore CUTOFF and hence SEARCH called in CUTOFF are executed $w$ or $w + 1$ times in total. Therefore we shall show that each execution of CUTOFF and SEARCH can be done in $O(k)$ time.

Let $N$ be the given even grid network, and assume that REDUCE had transformed $N$ into an even network $N'$ by deleting several edges in $C_3(e_0)$. Furthermore, suppose that we are going to execute CUTOFF$(N', e, B_1)$ for an edge $e \in E(B_1) \cap C_3(e_0)$. Rigorously speaking, $N'$ is not a grid network since the outer boundary $B_1$ or the inner boundary $B_1$ is no longer a rectangle. However we extend the definition of a rectilinear or box cut to such a network $N'$, as illustrated in Fig. 10. Then, similarly as in Lemmas 3.2 and 3.3, one can observe that Theorem 3.4 holds even for $N'$, that is, $N'$ satisfies the cut condition if and only if all rectilinear and box cuts of $N'$ have nonnegative margings.

We now define an “L-cut” as a cutset $C(X)$ such that $C(X) - e$ is a rectilinear or box cut of $G - e$. Furthermore we define $m'(e)$, $m'(q, r)$ and $m'_i(q)$ for $e \in E(B_1) \cap C_3(e_0)$, $e_q, e_r \in E(B_1)$ and $(t, t') \in P_i$ as follows:

$$m'(e) = \min\{m(X): e \in C(X) \text{ and } C(X) \text{ is a box or L-cut}\};$$

$$m'(q, r) = \min\{m(X): e_q, e_r \in C(X) \text{ and } C(X) \text{ is a rectilinear, box or L-cut}\};$$
and

\[ m'_i(q) = \text{MIN}\{m'(q, r): \text{index}(t) \leq r \leq b\}. \]

Then CUTOFF and SEARCH correctly works for the grid network \( N' \) even if \( m'(e) \), \( m'(q, r) \), and \( m'_i(q) \) are used in place of \( m(e) \), \( m(q, r) \), and \( m_i(q) \).

We now estimate the time needed to compute these values. Clearly \( m'(e) \) can be computed in \( O(k) \) time. Moreover one can compute in \( O(k) \) time values \( m'(q, r) \) for
a fixed edge $e_q \in E(B_i)$ and all edges $e_r \in E(B_i)$. Similarly as in Section 3, we update these values to those for the edges $e_{q-1}$ next to $e_q$ on $B_i$. During each execution of SEARCH the updating occurs $O(k)$ times, and hence one can execute all updating either in $O(k \log k)$ time by using the height-balanced binary tree or in $O(k)$ time by using the variable-priority queue [14]. Thus one can compute all $m_i'(q)$ needed in each execution of SEARCH total in $O(k)$ time.

We next consider the computation time of SEARCH necessary to find the pair $(t, t')$ in $Q$ such that $t$ first appears on the boundary $B_i$ counterclockwise starting from $u_0$. Clearly one can find the first pair $(t_1, t_1')$ in $O(b - \text{index}(t_1))$ time simply by checking whether pairs having terminals on vertices $v_{0}, v_{b-1}, \ldots, v_{\text{index}(t_1)}$: otherwise, $(t_2, t_2')$ would have been selected as $(t_1, t_1')$. Therefore one can find $(t_2, t_2')$ in $O(\text{index}(t_1) - \text{index}(t_2))$ time by checking pairs having terminals on vertices $t_1, v_{\text{index}(t_1) - 1}, \ldots, v_{\text{index}(t_2)}$. Repeating this process, one can find all $(t, t')$ in $O(k)$ time during each execution of SEARCH.

Thus each execution of SEARCH and hence CUTOFF can be completed in $O(k)$ time. Therefore the execution of REDUCE can be done in $O(kw)$ time, as we claim.

One more idea is necessary to make the algorithm run in $O(n)$ time. Let $w_i = |C_i(e)|$ for an edge $e$ on $B_i^l$, $1 \leq i \leq 4$. Assume that $w_1$ is minimum among $w_1$, $w_2$, $w_3$ and $w_4$, and $w_j$ is minimum among $w_2$, $w_3$ and $w_4$. As illustrated in Fig. 11, we apply REDUCE to an even grid network $N$ with respect to two appropriate semicuts $C_i(e)$ and $C_i(e')$, $e \in E(B_i^l)$ and $e' \in E(B_i^r)$, so that $N$ is decomposed into two L-shaped grids having all terminals only on the outer boundaries. This reduction can be done in $O(k(w_1 + w_j))$ time. One can easily observe $b_l(w_1 + w_j) = O(n)$. (Compare the area of the region bounded by $B_1$ and $B_2$ with the total area of two rectangles of size $b_1 \times w_1$ and $b_1 \times w_1$.) Clearly $b_2 \leq b_1$ and $k \leq b_1 + b_2$. These three equations imply $k(w_1 + w_j) = O(n)$. Hence the reduction above can be done in $O(n)$.

Fig. 11. Decomposition of $N$ into two L-shaped grids.
time. On the other hand one can find edge-disjoint paths in an L-shaped even grid having all terminals on the outer boundary in $O(n)$ time [10]. Thus we conclude that edge-disjoint paths in $N$ can be found in $O(n)$ time.

5. General grid networks

In this section let $N$ be a grid network that is not necessarily even. By definition $N$ satisfies the constraint (2) in the Introduction. One example is given in Fig. 2. We give a strong sufficient condition for $N$ to have edge-disjoint paths.

**Theorem 5.1.** A grid network $N$ has edge-disjoint paths if

$$m(X) \geq \begin{cases} 1, & \text{for every rectilinear cut } C(X), \\ 3, & \text{for every box cut } C(X). \end{cases}$$

**Proof.** Both $B_1$ and $B_2$ have even number of vertices of odd extended degrees. Add a dummy terminal to each of these vertices, and pair off them in the clockwise order around $B_1$ and $B_2$. Let $N'$ be the resulting even grid network. (The even network in Fig. 1 is indeed obtained from the noneven network in Fig. 2 in this way, where dummy terminals are 12, 13, ..., 21.) Clearly the margin $m'(X)$ of a cutset $C(X)$ in network $N'$ satisfies

$$m'(X) \geq \begin{cases} m(X) - 2, & \text{if } C(X) \text{ is a rectilinear cut}, \\ m(X) - 4, & \text{if } C(X) \text{ is a box cut}. \end{cases}$$

Since $N'$ is even, $m'(X)$ is an even integer. Therefore it follows from (4) and (5) that $m'(X) \geq 0$ for every rectilinear or box cut in $N'$. By Theorem 3.4, $N'$ and hence $N$ have edge-disjoint paths. 

Clearly one can check the condition above for a given grid network $N$ in $O(b_1)$ time, and actually find paths in $O(n)$ time.

**Acknowledgement**

We would like to thank Professors N. Saito and H. Takeda for fruitful discussions, and Professor J.S. Deogun and the referee for many suggestions to improve the presentation of the paper. This research is partly supported by Grant in Aid for Scientific Research of the Ministry of Education, Science, and Culture of Japan under a grant number: General Research (C) 62550253 (1987).
References


