ROUTING THROUGH A GENERALIZED SWITCHBOX

by

Michael Kaufmann / Kurt Mehlhorn

A 84/11

Fachbereich 10
Universität des Saarlandes
D-6600 Saarbrücken
West Germany

Also available as
SFB 17/1984

Abstract:

We present an algorithm for the routing problem for two-terminal nets in generalized switchboxes. A generalized switchbox is any subset R of the planar rectangular grid with no non-trivial holes, i.e. every finite face has exactly four incident vertices. A net is a pair of nodes of non-maximal degree on the boundary of R. A solution is a set of edge-disjoint paths, one for each net.

Our algorithm solves standard generalized switchbox routing problems in time $O(n(\log n)^2)$ where n is the number of vertices of R, i.e. it either finds a solution or indicates that there is none. A problem is standard if $deg(v) + ter(v)$ is even for all vertices v where deg(v) is the degree of v and ter(v) is the number of nets which have v as a terminal. For non-standard problems we can find a solution in time $O(n(\log n)^2 + |U|^2)$ where U is the set of vertices v with $deg(v) + ter(v)$ is odd.

I. Introduction

In this paper we solve the routing problem for two-terminal nets in generalized switchboxes. A generalized switchbox is any subset R of the planar rectangular grid without holes, i.e. all finite faces of R have exactly four incident edges. (cf. Figure 1). Let $B(R) := \{v;\ v \text{ node of } R \text{ and } v \text{ has degree} \le 3\}$ be the nodes of R which do not have maximal degree. Note that all nodes of B(R) are incident to the infinite face. A two-terminal net is an unordered pair of points in B(R). A generalized switchbox routing problem (GSRP) is given by a generalized switchbox R and a set $N = \{(s_i, t_i)\ ;\ 1 \le i \le m\}$ of nets. A solution to the problem is a set $P = \{p_i,\ 1 \le i \le m\}$ of paths such that

(1) p_i connects s_i and t_i for $1 \le i \le m$
(2) p_i and p_j are edge-disjoint for $i \neq j$.

In this paper we present an algorithm which solves standard generalized switchbox routing problems in time $O(n(\log n)^2)$ where n is the number of vertices of the routing region R. A routing problem is standard if $\deg(v) + \operatorname{ter}(v)$ is even for all nodes v where deg(v) is the degree of node v and ter(v) is the number of nets which have v as a terminal. We call $\deg(v) + \operatorname{ter}(v)$ the extended degree of node v. For non-standard GSRPs we do slightly worse. We show how to find a solution in time $O(n \log^2 n + |U|^2)$, where U is the set of vertices with odd extended degree.

A solution to a routing problem in the sense described above is usually called a solution in knock-knee mode. Note that a vertex v of R is used by either one wire or two wires which either go straight through v or bend in v (cf. Figure 2). Previous work on routing problems in kock-knee mode can be found in Preparata/Lipski, Frank, Mehlhorn/Preparata, Becker/Mehlhorn, Kramer/v. Leeuwen, and Brady/Brown. Preparata/Lipski solve the channel routing problem, Frank and Mehlhorn/Preparata solve the switchbox routing problem. A switchbox is a rectangular subset of the plane grid. The running time of their algorithm is O(nlog n) and O(ulog u) respectively where u is the circumference of the rectangle. Becker/Mehlhorn consider a more general problem than the one considered here. They consider arbitrary subsets of the planar grid (holes are allowed!!) and solve the routing problem in time $O(n^{3/2})$. Finally Brady/Brown consider the problem of layer assignment. They show that any layout in knock-knee mode can be wired using four conducting layers.

All papers mentioned above (except Brady/Brown) and also the present paper are based on a theorem of Okamura/Seymour on multi-commodity flow in planar graphs. We review their theorem in section 2. In section 3 we refine their theorem to the special case of standard generalized switchboxes. In section 4 we derive an algorithm for standard GSRPs and analyse its running time. In section 5 we deal with non-standard GSRPs.

2. The Theorem of Okamura/Seymour

Let $G = (V,E)$ be a graph and let N be a set of unordered pairs of vertices of G; $N = \{(s_i,t_i);\ 1 \le i \le m\}$. A cut is a subset $X \subseteq V$ of the vertices of G. The capacity of a cut X is the number of edges in E with exactly one end in X and the density of a cut X is the number of nets $(s,t) \in N$ with exactly one terminal in X, i.e.

$$\mathrm{cap}(X) = |\{e \in E;\ e = (a,b) \text{ and } a \in X,\ b \notin X\}|$$
$$\mathrm{dens}(X) = |\{(s,t) \in N;\quad s \in X,\ t \notin X\}|$$

We will also use

$CAP(x) = \{e \in E;\ e = (a,b) \text{ and } a \in X,\ b \notin X\}$ and

$dens(X_1,X_2) = \{(s,t) \in N;\ \ s \in X_1,\ t \in X_2\}$
for $X_1,X_2 \subseteq V,\ X_1 \cap X_2 = \emptyset$

Theorem (Okamura/Seymour): If G is planar and can be drawn in the plane such that $s_1,\ldots,s_m,t_1,\ldots,t_m$ are all on the boundary of the infinite region and $cap(X) - dens(X)$ is non-negative and even for all cuts $X \subseteq V$ then there are pairwise edge-disjoint paths $p_1,\ldots,p_m$ such that p_i connects s_i and t_i, $1 \le i \le m$.

Okamura/Seymour give a constructive proof of their theorem; their proof leads to the following algorithm which can be made to run in time $O(n^2)$ as shown by Becker/Mehlhorn.

Let $\hat{G}$ be an embedding of G with $s_1,\ldots,s_m,t_1,\ldots,t_m$ on the boundary of the infinite face. We may assume w.l.o.g. that $\hat{G}$ is 2-connected. Then the boundary of the infinite face consists of a circuit C which we regard as a subgraph of G. We say that a cut X is critical if X is connected, saturated, i.e. $cap(X) = dens(X)$, and $CAP(X)$ contains exactly two edges of C. Thus if X is critical then $C \mid (V(C) \cap X)$ and $C \mid (V(C) - X)$ are both paths.

We can now describe the algorithm.

let $e = (v,w)$ be an arbitrary edge on the boundary C of the infinite face of C;
if there is a critical cut X with $v \in X$, $w \notin X$
then let X be such a critical cut with
$|V(C) \cap X|$ minimal;
let$(s,t) \in N$ be a net with $s \in X$, $t \notin X$ such that the subpath of C from w to t not using v has minimal length; (cf. Figure 3)

remove edge e from G;
replace net (s,t) by the pair (s,v) and (w,t) of nets;
construct a solution for the reduced graph using the algorithm recursively and obtain the path for net (s,t) by connecting the paths for nets (s,v) and (w,t) by edge e.

<u>else</u> remove edge e from G and add net (v,w) to the set of nets;
construct a solution for the reduced graph and throw away the path for net (v,w)

<u>fi</u>

The correctness of this algorithm can be deduced from the paper of Okamura/Seymour; a proof can be found in Becker/Mehlhorn.

We close this section with a collection of simple observations. For a vertex $v \in V$ let deg(v) be the degree of v and let ter(v) be the number of nets in N which have v as a terminal. We call a routing problem (given as a planar graph and a set of nets) <u>standard</u> if the extended degree deg(v) + ter(v) is even for all $v \in V$. We call it <u>solvable</u> if it has a solution.

<u>Lemma 1:</u> Let $G = (V,E)$ be a planar graph and let N be a set of nets having their terminals on the boundary of the infinite face.

a) The routing problem (G,N) is standard <u>iff</u> cap(X) - dens(X) is even for every cut X.

b) A standard routing problem (G,N) is solvable <u>iff</u> no cut X is oversaturated, i.e. there is no cut with cap(X) < dens(X).

Proof: a) Let $X \subseteq V$ be arbitrary. We have

$$\mathrm{dens}(X) = \sum_{v \in X} \mathrm{ter}(v) - 2\,|\{(s,t);\ (s,t) \in N \text{ and } s,t \in X\}|$$

and

$$\mathrm{cap}(X) = \sum_{v \in X} \deg(v) - 2\,|\{(a,b);\ (a,b) \in E \text{ and } a,b \in X\}|$$

This proves a)

b) If (G,N) is solvable then there is clearly no oversaturated cut. Conversely, if $\mathrm{dens}(X) \le \mathrm{cap}(X)$ for every cut X then $\mathrm{cap}(X) - \mathrm{dens}(X)$ is non-negative and even by part a). Hence (G,N) is solvable by Okamura/Seymour. $\square$

3. Critical Cuts in Standard Generalized Switchboxes

Let R be a generalized switchbox. We use $C(R)$ to denote its boundary, i.e. the boundary of the infinite face. Let $B(R) = \{v \in C(R);\ \deg(v) \le 3\}$ and let $N \subseteq B(R) \times B(R)$ be a set of nets. We assume throughout this section that (R,N) is a standard problem, i.e. $\deg(v) - \mathrm{ter}(v)$ is even for all $v \in V$.

Our first goal is to show that nodes $v \in C(R)$ with $\mathrm{ter}(v) = \deg(v)$ are easily handled.

Lemma 2: Let $v \in B(R)$ be a node with $\deg(v) = \mathrm{ter}(v)$. Let (v,t_i), $1 \le i \le \mathrm{ter}(v)$, be the sets which have v as a terminal and let b_i, $1 \le i \le \mathrm{ter}(v)$, be the neighbors of v. The following transformations turn a solvable problem into a solvable problem.

1) If $\mathrm{ter}(v) = 1$ then delete v and replace net (v,t_1) by net (b_1,t_1)

2) If $\mathrm{ter}(v) = 2$ then let b_1,t_1,t_2,b_2 be the order in which these four points lie on circuit C; consecutive points may be equal. Delete node v and replace nets (v,t_1), (v,t_2) by (b_1,t_1) and (b_2,t_2).

3) If $ter(v) = 3$ then let b_1, b_3 be neighbors of v on circuit C. Let b_1, t_1, t_2, t_3, b_3 be the order in which these five points lie on circuit C; consecutive points may be equal. Delete node v and replace net (v, t_i) by (b_i, t_i), $1 \le i \le 3$.

Proof: We prove part 3) the other two cases being simpler. Consider a solution $p_1, \ldots, p_m$ for our routing problem. Assume w.l.o.g. that p_i is the path for net (v, t_i), $1 \le i \le 3$. We may assume w.l.o.g. that paths p_1, p_2, p_3 do not cross. Hence path p_i passes through vertex b_i for $1 \le i \le 3$. Thus the modified problem is solvable. $\square$

Lemma 1 allows us to simplify routing problems. In a simplified standard generalized switchbox routing problem (SSGS) there are no nodes v with $deg(1) = 1$, and all nodes v with $deg(v) = 2$ ($deg(v) = 3$) satisfy $ter(v) = 0$ ($ter(v) = 1$). Also nodes with $deg(v) = 4$ satisfy $ter(v) = 0$.
We will next characterize the form of critical cuts in SSGSs.
Let (R, N) be a solvable SSGS. We may index the vertices of R by integer coordinates. Let v be a left upper corner (i.e. $deg(v) = 2$ and the left and top neighbor of v do not exist) of R with maximal y-coordinate. Let w be the bottom neighbor of v (cf. Figure 4). Note that no vertex of R has y-coordinate larger than v.

We consider critical cuts X with $v \in X$ and $w \notin X$ if there are any. Among these cuts we select one with $|V(C) \cap X|$ minimal and among these cuts we select one with $|X|$ minimal. We denote this cut by X_o. One main goal of this section is to show that X_o has a very simple form. Its boundary consists of at most two line segments (see Lemma 4 for a precise statement). We start with several simple observations.

1) $R|X_o$ is connected. Otherwise we could take as X_o the connected component of $R|X_o$ containing v, a contradiction to the choice of X_o.

2) $R|X_o$ is a generalized switchbox. Assume otherwise. Let $X' \supseteq X_o$ be obtained from X_o by filling the holes. Then $v \in X'$, $w \notin X'$, $dens(X') = dens(X_o)$ and $cap(X') < cap(X_o)$. Thus X' is oversaturated and our routing problem is not solvable.

3) Let v and v' be the endpoints of the path $V(C) \cap X_o$. Then every node $x \in X_o - \{v,v'\}$ has degree ≥ 2 in $R|X_o$. Assume otherwise. Consider cut $X' = X_o - \{x\}$. Then $R|X'$ is still connected, $V(C) \cap X'$ is still a path and hence $x \notin V(C)$, $dens(X') \geq dens(X_o)$ and $cap(X') \leq cap(X_o)$, a contradiction to the choice of X_o.

Consider the edges in $CAP(X_o)$, i.e. the edges with exactly one endpoint in X_o. We can view the "cut" X_o as a polygonal line S intersecting exactly the edges in $CAP(X_o)$. Line S consists of straight line segments $s_1, s_2, \ldots, s_k$ where s_1 intersects the edge (v,w).

<u>Lemma 3:</u> Each segment s_i intersects an edge $e = (x,y) \in CAP(X_o)$ such that either x or y lies on the boundary C(R) of R.

<u>Proof:</u> The claim is certainly true for segments s_1 and s_k. Assume now that there is a segment s_i, $1 < i < k$, which cuts no edge incident to a node on the boundary. Assume w.l.o.g. that s_i is vertical and that X_o is to the right of s_i. Then s_{i-1} and s_{i+1} are horizontal.
<u>case 1:</u> either s_{i-1} or s_{i+1} extends to the right of s_i. Then we can move s_i one unit to the right and obtain a cut X' with dens $(X') = dens(X_o)$, $|X'| < |X_o|$, $cap(X') \leq cap(X_o)$ and $|V(C) \cap X'| = |V(C) \cap X_o|$, a contradiction to the minimality of X_o.

<u>case 2:</u> s_{i-1} and s_{i+1} extend to the left of s_i. Then we can move s_i one unit to the left and obtain a cut X' with dens(X') = dens(X_o) and cap(X') = cap(X)-2. Thus X' is oversaturated, a contradiction to our global assumption that we deal with a solvable problem. □

<u>Lemma 4:</u> Line S consists of at most two segments. In addition, if there are two segments then the angle $\measuredangle (s_1,s_2)$ is concave relative to X_o.

<u>Proof:</u> Assume first that angle $\measuredangle (s_1,s_2)$ is convex relative to X_o. (cf. Figure 5). Then k = 2. Since ter(v) = 0 and ter(x) = 1 for all other nodes $x \in X_o$ cut X_o cannot be saturated, a contradiction.

This shows that $k \geq 2$ implies that $\measuredangle (s_1,s_2)$ is concave relative to X_o. It remains to show that $k \leq 2$. Assume otherwise, i.e. $k \geq 3$. We have to distinguish two cases

<u>Case 1:</u> $\measuredangle(s_2, s_3)$ is convex relative to X_o. We know from the proof of lemma 3 that there are points in C(R) immediately to the right of s_2. Let a be the lowest such boundary point above s_3. Then either the point above or below a is also a boundary point.

<u>Case 1.1:</u> The point immediately below a is not a boundary point. Then the point above a is a boundary point; call it b. (cf. Figure 6)
We consider the two cuts as shown in figure 7. Note that cut X_2 exists since vertex a was chosen as the lowest boundary point to the right of s_2. We have

$$\mathrm{cap}(X) = \mathrm{cap}(X_1) + \mathrm{cap}\ (X_2)$$

and

$$\mathrm{dens}(X) = \mathrm{dens}(X_1) + \mathrm{dens}(X_2) - 2\ \mathrm{dens}(X_1,X_2)$$

since vertex a has degree 4 (if a had only degree 3 or less then R would not be biconnected) and hence ter(a) = 0.

$\leq \text{dens}(X_1) + \text{dens}(X_2)$

From $\text{cap}(X) = \text{dens}(X)$ and $\text{cap}(X_i) \geq \text{dens}(X_i)$ for $i = 1,2$ we conclude $\text{cap}(X_i) = \text{dens}(X_i)$ for $i = 1,2$. In particular, X_1 is saturated. This contradicts the minimality of X_o.

<u>Case 1.2 :</u> The point below a is a boundary point and hence s_3 cuts only one edge (cf. Figure 7). If the point above a is also a boundary point then we can certainly shorten cut X_o and still have a saturated cut, a contradiction. So let us assume that the point above a is not a boundary point. Let b be the boundary point which lies above a and is closest to a. Then the boundary C(R) either goes straight through b or bends in b.

<u>Case 1.2.1 :</u> The boundary C(R) goes straight through b. (cf. Figure 8). Then b must lie in the top row of R. We consider the cut X' obtained by moving s_2 one unit to the right. (cf. Figure 9). We have $\text{dens}(X') = \text{dens}(X_o)$ since $\text{ter}(a) = 0$ (note that $\text{deg}(a) = 4$). Also $\text{cap}(X') = \text{cap}(X_o)$ and hence X' is saturated. This contradicts the minimality of cut X_o.

<u>Case 1.2.2 :</u> The boundary C(R) bends in vertex b. (cf. Figure 10). Consider cuts X_1 and X_2 as indicated in Fig. 11. We have $\text{cap}(X_o) = \text{cap}(X_1) + \text{cap}(x_2)$ and $\text{dens}(X_o) = \text{dens}(X_1) + \text{dens}(X_2) - 2\,\text{dens}(X_1,X_2) \leq \text{dens}(X_1) + \text{dens}(X_2)$. Thus $\text{cap}(X_1) = \text{dens}(X_1)$, a contradiction to the minimality of X_o.

<u>Case 2 :</u> $\measuredangle(s_2,s_3)$ is concave relative to X_o. The proof of lemma 3 implies that there is a boundary point immediately to the left of segment s_2.

<u>Case 2.1 :</u> s_3 cuts at least two edges. Then the boundary points to the left of s_2 lie in h (≥ 1) segments as shown in Figure 12. Let ℓ_i be the number of vertices in the segment between a_i and b_i inclusive, $1 \leq i \leq h$. Note that

$\deg(a_i) = \deg(b_i) = 4$ and hence $ter(a_i) = ter(b_i) = 0$. We consider cuts $X_1,\dots,X_{h+1}$ as shown in Figure 14. We have

$$cap(X_o) = cap(X_1) + \dots + cap(X_{h+1}) + \sum_{i=1}^{h} (\ell_i - 2) + 2$$

since $\ell_i - 2$ horizontal edges are not cut anymore in the i-th segment and two vertical edges are not cut anymore. These edges are indicated as dashed lines in Figure 13. Also

$$dens(X_o) \le dens(X_1) + \dots + dens(X_{h+1}) + \sum_{i=1}^{h} (\ell_i - 2)$$

since every net which goes across cut X also goes across one of the cuts X_i or has a terminal in one of the vertical segments between a_i and b_i. Since $cap(X_o) = dens(X_o)$ and $cap(X_i) \ge dens(X_i)$, $1 \le i \le h+1$, we conclude

$$\begin{aligned} cap(X_1) + \dots + cap(X_{h+1}) + \sum_{i=1}^{h} (\ell_i - 2) + 2 \\ &= cap(X_o) \\ &= dens(X_o) \\ &\le dens(X_1) + \dots + dens(X_{h+1}) + \sum_{i=1}^{h} (\ell_i - 2) \\ &\le cap(X_1) + \dots cap(X_{h+1}) + \sum_{i=1}^{h} (\ell_i - 2), \end{aligned}$$

a contradiction.

Case 2.2 : S_3 cuts exactly one edge. Then the situation is as shown in Figure 14. Let ℓ_i be the number of vertices between and including a_i and b_i, $1 \le i \le h-1$, let ℓ_h be the number of vertices below and including a_h and above s_3. Consider cuts $X_1,\dots,X_h$ as shown in Figure 15. We have

$$cap(X_o) = cap(X_1) + \dots + cap(X_h) + \sum_{i=1}^{h-1} (\ell_i - 2) + (\ell_h - 1) + 2$$

and

$$dens(X_o) \le dens(X_1) + \dots + dens(X_h) + \sum_{i=1}^{h-1} (\ell_i - 2) + \ell_h - 1$$

As in case 2.1 we can now derive a contradiction. This finishes the case analysis and proves lemma 4. □

Lemma 4 is very crucial for the efficiency of our algorithm. It completely characterizes the form of the minimal critical cuts X_o through edge (v,w).

4. The Algorithm

Let R be a generalized switchbox with n vertices and let N be a set of nets. Throughout this section we assume that (R,N) is a standard problem. The goal of this section is to describe an algorithm which solves any standard generalized switchbox routing problem in time $O(n(\log n)^2)$.

The algorithm is a special case of the general multicommodity flow algorithm outlined in section 2.
It derives its speed from the clever use of the characterization of minimal critical cuts derived in section 3.
The algorithm processes the routing region R row by row starting at the top row. In every step it considers a left upper corner in the top row, say v, and eliminates the vertical edge (v,w) incident to v as described in section 2. There are two main tasks which we have to solve (efficiently).

(1) find the minimal critical cut X_o through edge (v,w), if there is any

(2) choose the appropriate net to route across cut X_o.

We use two data structures to solve these tasks efficiently. The first data structure is a range tree for the set of nets and is global to the algorithm. The second data structure is a priority queue for the free capacities of the cuts through edge (v,w) and is local to each row of the routing region. We assume that the vertices on the boundary

$C(R)$ of the routing region are numbered in clock-wise order by the integers in range $[1..M]$.
As the algorithm proceeds vertices in $C(R)$ are deleted (always a left upper corner) and new nodes become boundary nodes. The new boundary vertices inherit the number from deleted vertices as shown in Figure 16. In this way the numbering of the boundary vertices remains in increasing clockwise order. However, adjacent boundary vertices are not necessarily numbered by consecutive integers. From now on we identify nodes in $C(R)$ with their number.

A net is represented as a pair of integers, namely by the pair of numbers associated with its terminals. The set $N = \{(s_i,t_i);\ s_i \le t_i\ ,\ 1 \le i \le m\}$ of nets is stored in a range tree. Range trees were introduced by Lueker and Willard; see also Mehlhorn, section VII.2.2 . We briefly review range trees. Range trees consist of a primary tree and a set of secondary trees, one for each node of the primary tree.

In our case the primary tree is a <u>static</u> search tree for integers $1,\dots,M$ of depth $O(\log M) = O(\log n)$. Let v be a node of the primary tree and let $NL(v) = \{(s,t) \in N;$ the leaf labelled s is a descendant of $v\}$. The secondary tree $ST(v)$ associated with node v is a balanced tree (AVL-tree, $BB[\alpha]$-tree, or ...) for the ordered multi-set $\{t;\ (s,t) \in NL(v)\}$. In every node w of a secondary tree we store two auxiliary fields: the first field contains the number of leaf descendants of w and the second field contains the maximal s such that net $(s,t) \in N$ is stored in that secondary tree and the leaf t is a descendant of w. It is clear that a range tree requires space $O(m \log M) = O(n \log n)$ since every net belongs to $O(\log M)$ node lists. It supports the following operations in time $O(\log n)^2)$.

1) Insert a net into N or delete a net from N

2) Given a,b,c,d find nets $(s,t) \in N$ and $(s',t') \in N$ with $a \le s,s' \le b$, $c \le t,t' \le d$ and t maximal or s' maximal respectively. These nets can be found as follows: Consider the search paths for a and b in the primary tree and let C_{max} to be the roots of the maximal subtrees of the primary tree between these paths. Then every net $(s,t) \in N$ with $a \le s \le b$ belongs to NL(v) for exactly one node $v \in C_{max}$. Also $|C_{max}| = O(\log M)$. For every node $v \in C_{max}$ we search for C in the secondary tree ST(v) and find the maximal t(v) and s'(v) such that $c \le t(v) \le d$, $c \le t'(v) \le d$ and $(s(v),t(v)) \in NL(v)$ and $(s'(v),t'(v)) \in NL(v)$. In order to find t(v) we only have to inspect the leaf immediately to the left of the search path for d and in order to find s'(v) we have to inspect the auxiliary fields of the nodes between the search paths to c and d. Finally comparing $(s(v),t(v))$ and $(s'(v),t'(v))$ for all $v \in C_{max}$ allows us to find the desired nets (s,t) and (s',t').

3) Given $a < b$ find the number of nets $(s,t) \in N$ with either $a \le s \le b < t$ or $s < a \le t \le b$. Let $n_1 = |\{(s,t);\ a \le s \le b < t\}|$ and $n_2 = |\{(s,t);\ s < a \le t \le b\}|$. We can determine n_1 as follows; n_2 is determined similarly. Define C_{max} as above. For every node $v \in C_{max}$ compute $|\{(s,t) \in NL(v);\ b < t\}|$ in time $O(\log n)$ by a search in ST(v) using the auxiliary information associated with the nodes.

The local data structures for the rows will be described below. We give the algorithm first.

(1) initialize the range tree for the set N of nets
(2) <u>while</u> routing region non-empty
(3) <u>do</u> consider a top row of the routing region;
(4) initialize the local data structure for the current row;

```
        while  row non-empty
        do  let v be the left corner of the row, let w be
            its bottom neighbor and let x be its right
            neighbor;
            if  ter(v) = deg(v)
            then  route as given by lemma 2 and delete v;
                  update data structures
            else  find minimal critical cut X through
                  edge (v,w);
                  if  this cut does not exist
                  then  delete node v, add net (x,w)
                        and update data structures
                  else  find net (s,t) to be routed across
                        cut X_o, s ∈ X_o, t ∉ X_o; delete
                        vertex v;
                        delete (s,t) from the set of
                        nets and add nets (x,s) and (w,t);
                        update data structures
                  fi
            fi ;
            split routing region if it is not biconnected
            anymore;
        od
od
```

We will next describe the local data structure for each row. Let L be the length of the top-row. We consider cuts consisting of one horizontal segment and one vertical segment or of only a horizontal segment. Let X_i be the cut where the horizontal segment intersects exactly i edges of the routing region. (cf. Figure 17). For every i let

$$\text{fcap}(X_i) = \text{cap}(X_i) - \text{dens}(X_i)$$

be the free capacity of cut X_i. We have to execute the following operations on $\text{fcap}(X_i)$, $1 \le i \le L$.

(1) compute $fcap(X_i)$, $1 \leq i \leq L$ in order to initialize the local data structure in line (4)

(2) find the maximal i with $fcap(X_i) = 0$ in order to find the minimal critical cut X_o through edge (v,w) in line (10).

(3) decrease $fcap(X_i)$ by two for $a \leq i \leq b$ in order to update the local data structure in lines (9), (13) and (18).

We show first how to do the first task in time $O(L(\log n)^2)$. Consider cut X_i. We know that $X_i \cap V(C)$ is a path and hence the numbers of vertices in $X_i \cap V(C)$ form an interval $[h,j]$ with $h < j$ or two intervals $[h,M]$, $[1,j]$ with $h > j$. Note that h is the number of vertex v. The integer j is easily found by computing in a preprocessing step for every vertex U of R the highest vertex below u which lies on C(R). The capacity of cut X_i is now readily computed in time O(1) by adding the lengths of its constituting segments. The density of cut X_i is computed in time $O((\log n)^2)$ using the third property of range trees derived above.

It remains to show how to solve the other two tasks. We use priority queues with updates as described in Galil/Naamad; see also Mehlhorn, section IV.9.1. They allow us to perform these tasks in time $O(\log n)$ each.

We will next discuss the lines of our algorithm in more detail. Lines (1) and (4) were already described. In line (8) we route as given by lemma 2. Let (v,t_1) and (v,t_2) be the two nets having v as a terminal with v,t_1,t_2 in clockwise order on C(R). Let i_o be maximal such that t_1 and t_2 both belong to X_i for $i \leq i_o$. Then $fcap(X_i)$ decreases by two for $i \leq i_o$. We also have to delete two nets from N and add two other nets. Thus the cost of line (8) is $O(\log n + (\log n)^2) = O((\log n)^2)$.

Line (10) takes time O(log n) by property (2) of the local data structure.

In line (12) we have to add one net to N and to reduce $fcap(X_i)$ by two for all i.

In line (13) we first have to find the net (s,t) which has to be routed across and cut X_o, i.e. $s \in X_o$, $t \notin X_o$ and t is as close to w as possible. Since $X_o \cap V(C)$ is a path the boundary nodes in X_o form an interval [h,j] with $h < j$ or two intervals [h,M] and [1,j] with $h > j$. In the former case net (s,t) is either the net (s',t') with $s' < h' \le t' \le j$ and s' maximal or the net (s",t") with $h \le s'' \le j \le t''$ and t" maximal. In the latter case the net (s,t) is either the net (s',t') with $s' < h \le t' \le M$ and s' maximal or the net (s",t") with $1 \le s'' \le h < t'' < j$ and t" maximal.

In line (16) we have to delete one net from N and add two other nets for a cost of $O((\log n)^2)$. In line (17) we have to change $fcap(X_i)$ for some cuts X_i. Let (s,t) be the net to be routed across X_o. Let i_o and i_1 be such that $s,t \notin X_i$ for $i \le i_o$ and $s,t \in X_i$ for $i \ge i_1$. Then $fcap(X_i)$ decreases by two for $i \le i_o$ and $i \ge i_1$. This change requires time O(log n).

We finally have to discuss line (20). Let y be the "diagonal" neighbor of vertex v. (cf. Figure 17). Then y (and only y) may become an articulation point by the removal of v. Vertex y becomes an articulation point if y belongs to C(R) before the removal of v, i.e. if y was numbered prior to the removal of v. Thus it is easy to test whether the routing region has to be split.

We split the routing region by finding the nets (s_i,t_i) which have to go through y using property (3) of the range tree and by replacing them by nets $(s_i,y),(y,t_i)$.

We then apply the algorithm separately to both parts. It is important to observe that we can use the <u>same</u> global data structure for both parts and that we can continue to process the current row, using the current local data structure.

This concludes the description of the algorithm and its data structures. The analysis of the running time is also easily completed at this point. All lines except line (4) take time $O((\log n)^2)$ and eliminate one vertex. Line (4) takes time $O(L(\log n)^2)$ where L is the length of the current row; i.e. time $O((\log n)^2)$ per vertex. Thus total running time is $O(n(\log n)^2)$. We summarize in

<u>Theorem 1:</u> Let (R,N) be a standard generalized switch-box routing problem with a routing region of n vertices. Then a solution (if there is one) can be constructed in time $O(n(\log n)^2)$.

5. Non-standard Routing Problems

This section is devoted to non-standard routing problems. We show how to find efficiently a solution for a non-standard GSRP if there is one.

We review the next two basic lemmas from the paper of Becker/Mehlhorn; the proofs can be found there.

<u>Lemma 5:</u> Let (R,N) be a non-standard GSRP which has a solution. Then there is a solvable standard GSRP(R,N') where $N' = N \cup P$ and P is a pairing of $U = \{v;\ v \text{ has odd extended degree}\}$

We call (R,N') a standard extension of (R,N).
Our extension is based on the concepts of <u>U-minimal cut</u> and <u>canonical extension</u>.

Let X be a saturated cut and let $u_1, u_2, \ldots, u_{2k}$ be the clockwise ordering of $X \cap U$. The cut X is U-minimal if $X \cap U \neq \emptyset$ and there is no simple saturated cut Y with $Y \cap U = \{u_i, u_{i+1}, \ldots, u_j\}$ with $1 < i < j < 2k$. The canonical extension of (R,N) with respect to X is obtained by adding nets (u_{2i-1}, u_{2i}), $1 \leq i \leq k$. Note that adding these nets will make the extended degrees of all vertices in X even.

<u>Lemma 6:</u> Given a solvable non-standard GSRP.
An iterative application of canonical extension with respect to U-minimal cuts leads to a solvable standard GSRP.

Lemma 6 leads to the following algorithm for turning a non-standard problem into a standard problem.

```
U_o  ← {v; extended degree of v is odd}
U ← U_o
while  U ≠ ∅  do
         if  there is an oversaturated cut
             then terminate and declare that the problem
                  has no solution
         fi
         let X be a U-minimal cut (X = V is possible)
         construct the canonical extension
         U ← U - X
od
```

Becker/ Mehlhorn showed how to implement this algorithm in time $O(bn + |U_o|^2) = O(bn)$ where b is the number of vertices on the boundary of the infinite face. Their algorithm works for arbitrary planar graphs where every interior node has even degree. The algorithm consists of two phases.

1) In phase one the free capacity of all cuts X is determined which can conceivably become U-minimal during

the extension of the algorithm. This phase takes $O(bn)$ and builds up a data structure of size $O(|U_o|^2)$ to be used in the second phase.

2) In phase two the algorithm above is used to construct the standard extension. Phase two takes time $O(|U_o|^2)$.

We will show how to execute phase one in time $O(n(\log n)^2)$ in our case. This will give an $O(n(\log n)^2 + |U_o|^2)$ algorithm for solving non-standard problems.

The main idea for the improved running time is the following: We may assume w.l.o.g. that U-minimal cuts have a very restricted form. Let X_o be a U-minimal cut. As in section 3 we can view X_o as a polygonal line S intersecting exactly the edges in $CAP(X_o)$. Line S consists of several straight line segments. We claim that two suffice.

<u>Lemma 7:</u> Let (R,N) be a solvable generalized switchbox routing problem with U as its set of vertices of odd extended degree. Then there is a U-minimal cut X_o consisting of at most two straight-line segments.

<u>Proof:</u> If V is a U-minimal cut then the claim is certainly true. Assume otherwise. Choose an U-minimal cut X_o consisting of straight line segments $s_1,\dots,s_k$ with k minimal. Note that $\emptyset \neq X_o \cap U \neq U$ since V is not U-minimal. If $k \leq 2$ then we are done.

So let us assume finally that $k \geq 3$. We may assume w.l.o.g. that s_1 is horizontal and the left end of s_1 intersects the boundary of R. Then s_2 intersects an edge of R whose left endpoint lies on the boundary of R. As in the proof of Lemma 4 we distinguish two cases.

Case A: s_3 runs to the right as seen from the lower endpoint of s_2.
Case Aa: extending s_3 for one segment of the left does not intersect a boundary edge (cf. Figure 18). Then the boundary points to the left of s_2 lie in h (≥ 1) segments as shown in Figure 18.
Let ℓ_i be the number of vertices in the segment between a_i and b_i inclusive, $1 \leq i \leq h$. Note that $\deg(a_i) = \deg(b_i) = 4$ and hence $\mathrm{ter}(a_i) = \mathrm{ter}(b_i) = 0$.
We consider cuts $X_1,\ldots,X_{h+1}$ as shown in Figure 19. Let o_i be the number of vertices of odd extended degree in the segment between a_i and b_i. We have

$$\mathrm{cap}(X_o) = \mathrm{cap}(X_1) + \ldots + \mathrm{cap}(X_{h+1}) + \sum_{i=1}^{h} (\ell_i - 2)$$

and

$$\mathrm{dens}(X_o) \leq \mathrm{dens}(X_1) + \ldots + \mathrm{dens}(X_{h+1}) + \sum_{i=1}^{h} (\ell_i - o_i - 2)$$

Since $\mathrm{cap}(X_o) = \mathrm{dens}(X_o)$ and $\mathrm{cap}(X_i) \geq \mathrm{dens}(X_i)$ for all i (we deal with a solvable problem) we conclude that $o_i = 0$ for all i, $1 \leq i \leq h$, and $\mathrm{cap}(X_i) = \mathrm{dens}(X_i)$ for $1 \leq i \leq h+1$. Since $o_i = 0$ for all i we conclude further that $U \cap X_o = (U \cap X_1) \cup \ldots \cup (U \cap X_{h+1})$ and hence one of the cuts X_i is U-minimal. This contradicts the choice of cut X_o.

Cases Ab ($\equiv$ not case Aa) and B (not case A) are similar and are left to the reader. □

Lemma 7 tells us that we only need to consider cuts with at most one bend when searching for U-minimal cuts. Let $e = (x,y)$ be an edge on the boundary of R and let $\ell(e)$ be the length of a cut through e which consists of a single straight line segment.
Clearly $\Sigma\, \ell(e) \leq O(n)$ where the sum is over all edges on the boundary of R. Also there are only $2\,\ell(e)$ cuts through

e with exactly one bend. Hence only $O(n)$ cuts must be considered as candidates for U-minimal cuts. For every single cut we can compute its free capacity in time $O(\log n)^2)$ as shown in section 4. Thus time $O(n(\log n)^2)$ suffices to compute the information required for the second stage of the algorithm in Becker/ Mehlhorn).
We summarize in

<u>Theorem 2:</u> Non-standard routing problems with n vertices and U vertices of odd extended degree can be solved in time $O(n(\log n)^2) + |U|^2)$.

<u>Proof:</u> By the discussion above one can extend the non-standard problem to a standard problem in time $O(|U|^2 + n(\log n)^2)$. The standard extension can be solved in time $O(n(\log n)^2)$ by theorem 1.

<u>Conclusion:</u> We exhibit a routing algorithm for two-terminal nets in generalized switchboxes. The algorithm runs in time $O(n(\log n)^2)$ and finds a solution -if there is one- in the case of standard problems. Several open questions remain.

1) Can the running time be improved?

2) Can we also solve non-standard problems optimally in time $O(n(\log n)^2)$?

3) Can one extend the result to more general routing regions and/or multiterminal nets?

6. References

1. M. Becker/ K. Mehlhorn "Routing and Edge-disjoint paths in planar graphs"
Technical report, FB 10, Univers. des Saarlandes
August 1984

2. M. Brady/ D. Brown "VLSI Routing: Four Layers Suffice"
MIT VLSI conference, 1984

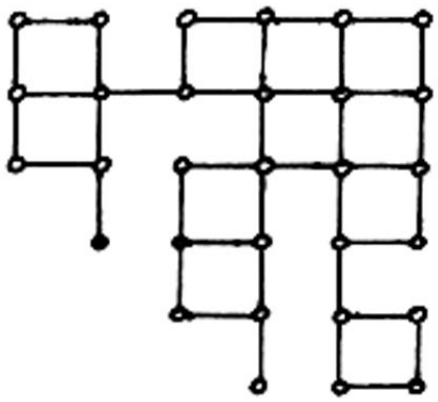
3. A. Frank "Disjoint Paths in Rectilinear Grids"
Combinatorica 2,4 (1982), 361-371

4. M.R. Kramer/ J. van Leeuwen "Wire Routing is NP-complete"
Technical Report RUU-CS-82-4, 1982, Utrecht

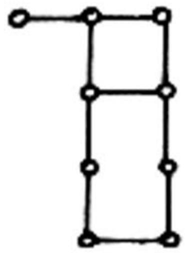
5. K. Mehlhorn/ F. Preparata "Routing Through a Rectangle"
Technical Report, 1983

6. H. Okamura, P.D. Seymour "Multicommodity flows in planar graphs"
Journal of Combinatorial Theory, Series B, 1981, 75-81

7. F. Preparata, W. Lipski "Three Layers are Enough"
23rd FOCS 1982, 350-357



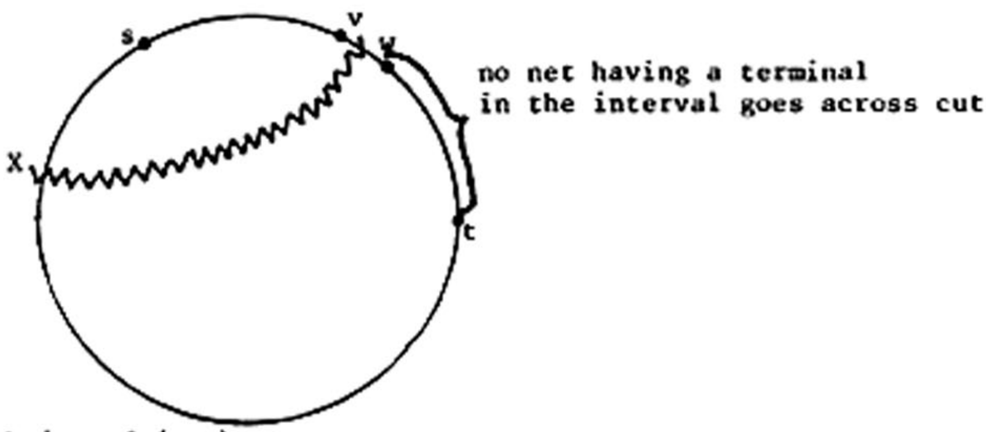
A generalized switchbox



Not a generalized switchbox

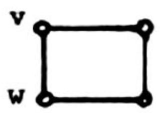
- Figure 1 -

- Figure 2 -



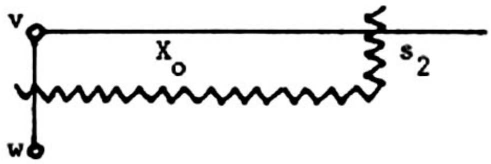
choice of (s,t)

- Figure 3 -



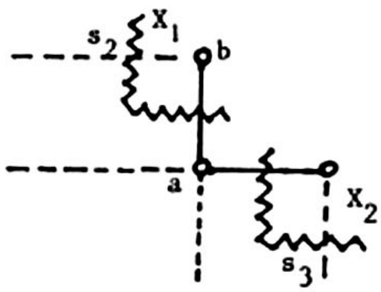
points v and w

- Figure 4 -



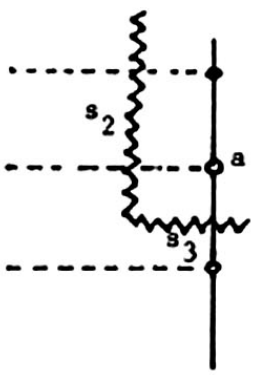
$\measuredangle\ (s_1,s_2)$ is convex and $k = 2$

- Figure 5 -

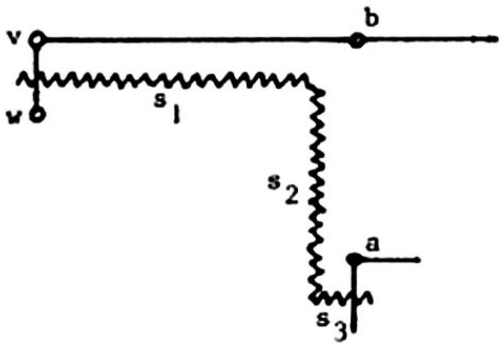


the boundary $C(R)$ is shown solid

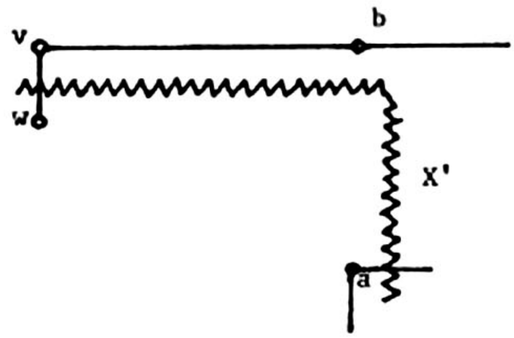
- Figure 6 -



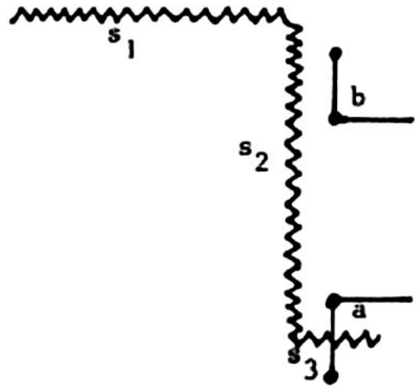
- Figure 7 -



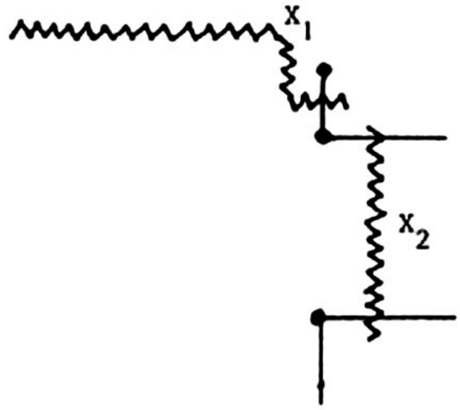
- Figure 8 -



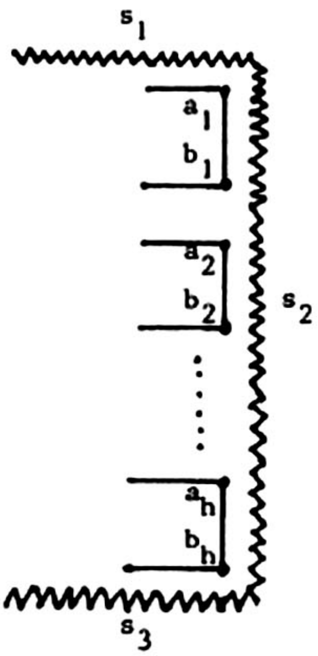
- Figure 9 -



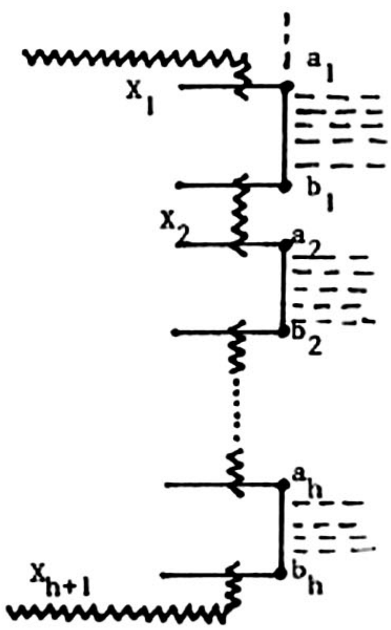
- Figure 10 -



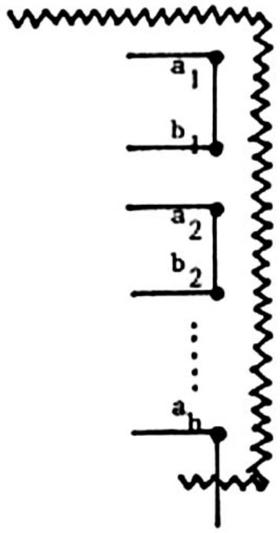
- Figure 11 -



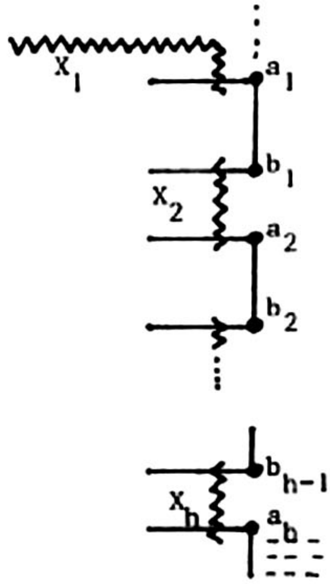
- Figure 12 -



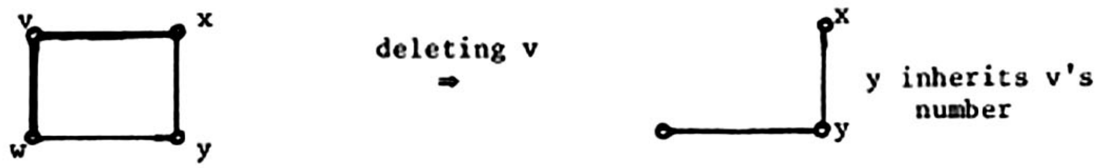
- Figure 13 -



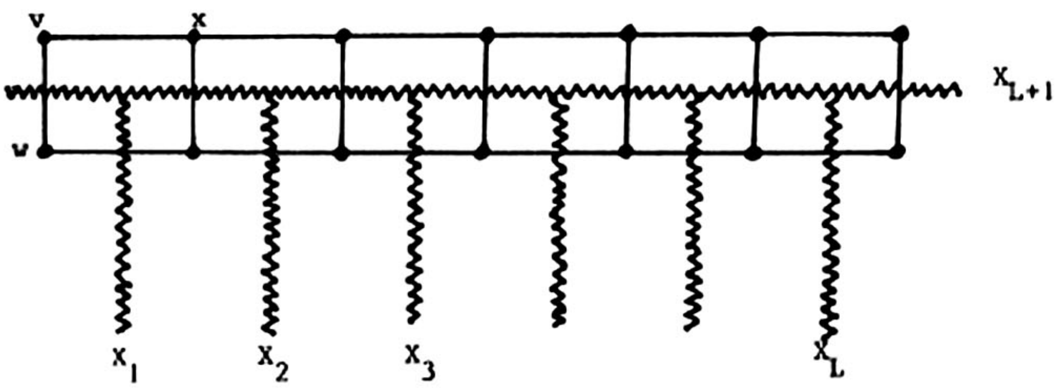
- Figure 14 -



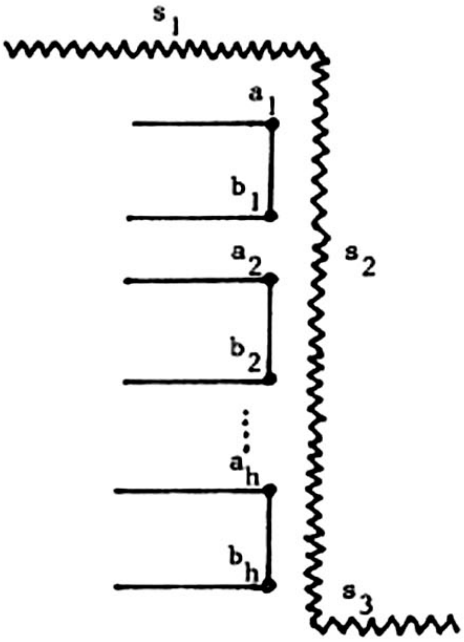
- Figure 15 -



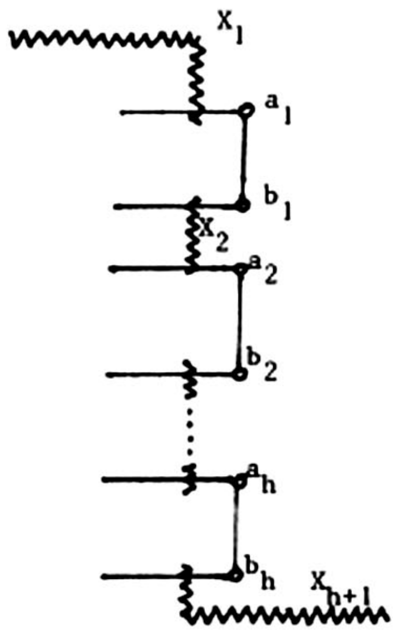
- Figure 16 - How numbers are inherited



- Figure 17 - The cuts X_i, $1 \le i \le L$



- Figure 18 -



- Figure 19 -