ON A ROUTING AREA DECOMPOSITION PROBLEM IN BUILDING-BLOCK LAYOUT DESIGN*†

Yang Cai and Martin D. F. Wong

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188

TR-91-32 October 1991

ABSTRACT

We study in this paper the problem of minimizing the number of L-shaped channels in routing area decomposition for building-block layouts. Given a building-block layout of rectangular modules, the routing area is to be decomposed into straight and L-shaped channels and routed in a certain order. Since (straight) channels are easier to route and channel routers usually produce near optimal results, it is desirable to minimize the number of L-shaped channels used in such a decomposition. We present an algorithm for minimizing the number of L-shaped channels used in the routing area decomposition. Our algorithm is based on a careful study of the structure of layouts of rectangular modules and a transformation of the original problem to a graph theoretical problem. For examples of up to 136 channels, our algorithm took less than one tenth of a second on a SUN SPARCstation 1 to finish the computation and obtained up to 29% reduction in the number of L-shaped channels over the results produced by the algorithm in [9].

Keywords: Computer-aided design, Channel routing, Building-block layout, Routing region definition and ordering, L-shaped channel.

---

* This work was partially supported by the National Science Foundation under grant MIP-8909586, by an IBM Faculty Development Award, and by an ACM SIGDA scholarship.

† A preliminary version of this paper was presented at the 28th Design Automation Conference [6].

1 Introduction

Detailed routing for VLSI building-block layout design is known to be a difficult problem [16, 18, 21, 24]. In order to reduce its complexity, routing is typically done in a hierarchical fashion [17, 20]. The routing area is to be decomposed into small regions which are then routed independently in a certain order such that whenever a region is being routed, its dimensions can be adjusted without destroying previously routed regions. In this case, routing completion in a single pass is guaranteed and there is no need for rip-up and reroute. This kind of routing area decomposition can be easily obtained for a special class of layouts called *slicing structures* [19]. However, this is not always possible for general layouts if we insist that the regions be rectangular in shape. For example, consider the layout shown in Figure 1, and suppose that the regions A, B, C, and D are all to be routed as straight channels. Clearly, A must be routed before B, for otherwise B is not completely defined because the positions of the terminals on the edge cd are not known at this time. For similar reasons, B must be routed before C, C must be routed before D, which in turn must be routed before A. Thus we have a *cyclic channel-routing precedence constraint*, which rules out the existence of a desired ordering of the routing regions.

Cyclic channel-routing precedence constraints can be avoided by converting the layout into a slicing structure [8, 17]. However, the conversion of a non-slicing tructure into a slicing structure usually result in an inferior placement. On the other hand, it is also possible to break the cyclic channel-routing precedence constraints without modifying the placement by the introduction of switchboxes [5, 20]. Again consider the layout shown in Figure 1, we can estimate the height of region A and fix the positions of the intersecting points of edge cd with the topological wiring paths generated in the global routing phase. These intersecting points are called *junction terminals*. Regions B, C, D can now be routed as straight channels in that order. After that, region A becomes a switchbox, having fixed width and fixed terminals on three of its four sides. It can be routed by a switchbox router [15, 24]. The drawback of this approach is that because the width of a switchbox is fixed before routing and cannot be expanded, routing completion is not guaranteed, and in some cases many iterations of rip-up and reroute are necessary in order to complete the routing.

In view of this difficulty, a new decomposition scheme was proposed in [9], in which some regions are combined to form larger regions called *L-shaped channels*. To see how the introduction of L-shaped channels can help to break cyclic channel-routing precedence constraints, consider the example shown in Figure 2, which is a new decomposition of the

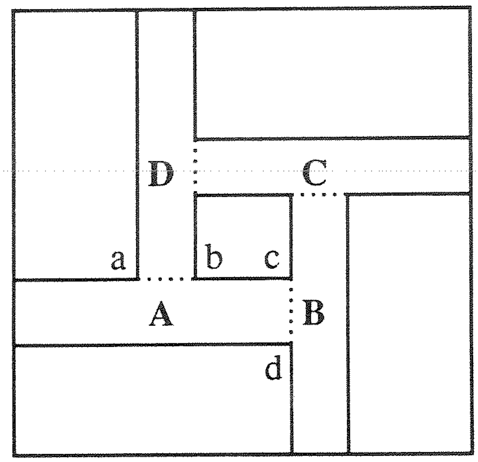


Figure 1: Cyclic channel-routing precedence constraint

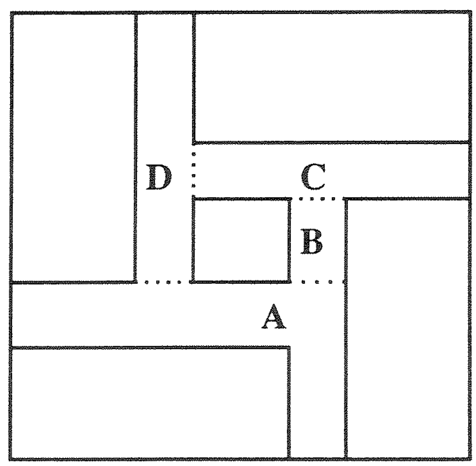


Figure 2: L-shaped channels

routing area for the layout shown in Figure 1. By making region A as an L-shaped channel, there is no cyclic channel-routing precedence constraint and the regions can be routed in the following order: B, C, D, A. Note that if the regions are routed in this order, then whenever a region is being routed, it can be expanded or contracted without destroying previously routed regions. Such an ordering of (straight) channels and L-shaped channels is called a *feasible channel ordering* of the layout. It is shown in [9] that if all the modules in the layout are rectangular in shape, then there always exists a way of decomposing the routing area into a feasible channel ordering of channels and L-shaped channels. The existence of a feasible channel ordering guarantees routing completion without rip-up and reroute.

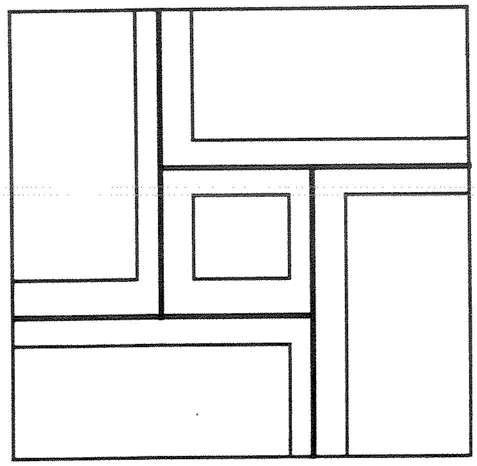
The channel routing problem has been extensively studied in the past two decades and its nature is well understood [10, 14, 18]. Many existing channel routers can achieve widths that are within one or two tracks of the optimal value [2, 10, 11, 23, 25]. On the other hand, the routing of L-shaped channels is a relatively new subject, its nature is not well understood and existing L-shaped channel router [7] may produce results which are inferior to the results produced by a channel router. Hence it is beneficial to use as few L-shaped channels as possible in decomposing the routing area. The problem of minimizing the number of L-shaped channels used appears to be quite difficult. In fact, it was conjectured to be NP-hard in [9]. The authors of [9] also proposed a heuristic method for minimizing the number of L-shaped channels used. However, their method is based only on local informations and hence could produce poor results for large layouts.

We present in this paper an efficient algorithm for computing a feasible channel ordering for a given layout of rectangular modules using as few L-shaped channels as possible. We use the novel method of maintaining global information and use them to correct "mistakes" made in the earlier stages of the algorithm. Our algorithm is based on a careful study of the structure of such layouts and a transformation of the original problem to a graph theoretical problem. The worst case time complexity of our algorithm is $O(n^2)$, where n is the number of modules in the given layout, and it runs extremely fast for practical examples. In particular, it took less than one tenth of a second of CPU time on a SUN SPARCstation 1 to finish the computation for examples of up to 136 channels. It produced results which are up to 29% better than that produced by the algorithm in [9].

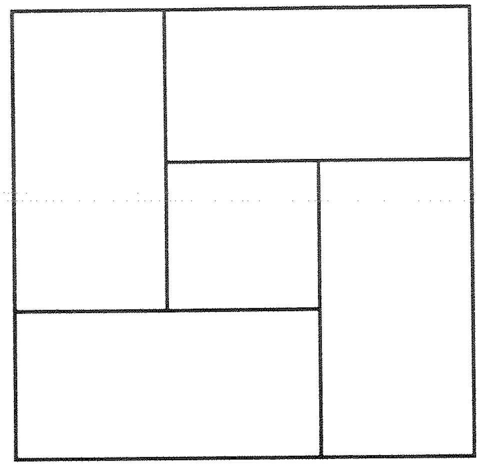
The rest of this paper is organized as follows. Terminologies, notations and preliminary results are introduced in Section 2. Section 3 reviews previous works done on this problem. Section 4 shows how to modify a feasible channel ordering of a layout to obtain a new feasible channel ordering with fewer L-shaped channels by changing some L-shaped channels into straight channels. Section 5 describes the construction of feasible channel orderings to be used as initial solutions. Section 6 presents our algorithm. Experimental results are presented in Section 7. Finally, we conclude the paper in Section 8.

2 Preliminaries

Given a layout of rectangular modules, we represent the routing regions between adjacent modules by line segments as illustrated in Figure 3(a). The diagram consisting of these line segments is shown in Figure 3(b). Such a diagram is called a *configuration*. In general,



(a) A layout



(b) Its configuration

Figure 3: A layout and its configuration

a configuration need not be rectangular in shape, it can be any rectilinear polygon which is convex in the Manhattan sense. [1] A configuration is said to be *trivial* if and only if it contains no line segments (except its boundary). An efficient algorithm for constructing a configuration from a given layout of rectangular modules were described in [8]. We assume throughout this paper that there is no empty room in the configuration. [2]

A *maximal* (*straight*) *line segment* of a configuration C is an internal straight line segment of C which cannot be further extended on either direction. The *limits of a straight line segment* of C are defined to be the endpoints of the maximal line segment that contains it. The *limits of an L-shaped line segment* of C is defined to be the two endpoints of the maximal line segments of C which contains the two straight line segments that form the L-shaped line segment, such that the line segments between the corner of the L-shaped line segment and these endpoints contain the two straight line segments of the L-shaped segment. In Figure 4, the limits of the L-shaped line segment L_3 are p and r, while its endpoints are c and r.

Given a configuration C, a *cut* of C is either a straight (horizontal or vertical) line segment of C, or an L-shaped line segment of C, cutting along which separates C into two line disjoint subconfigurations. It is easy to see that a straight or L-shaped line segment of

[1]A polygon P is *convex in the Manhattan sense* if and only if any horizontal or vertical line segment joining two points of P lies completely in P.

[2]An *empty room* in a configuration is a minimal closed region of the configuration such that in the layout corresponding to the configuration there is no block contained in the region. See [4] for an algorithm for removing empty rooms from a configuration.

C is a separating line segment, *i.e.*, a cut of C, if and only if both of its endpoints are on the boundary of C. Furthermore, the two subconfigurations resulting from cutting a cut of C are convex in the Manhattan sense if C is convex in the Manhattan sense. A cut is an *S-cut* if it is a straight line segment, otherwise it is an *L-cut*. An L-cut of C is said to be a *definite L-cut* if its corner is an "L"-type junction of C (it is so called because we will need an L-cut for each "L"-type junction of C). Since all modules in the layout are rectangular, we have

Lemma 2.1 *Every nontrivial configuration C contains a cut.*

An *cut sequence* of a configuration C is a sequence $\pi = \pi_1\pi_2\ldots\pi_n$, such that:

- π_1 is a cut of $C = C[\epsilon]$, where ϵ is the null sequence, and $C[\pi_1]$ is the set of line disjoint configurations obtained by cutting $C[\epsilon]$ along π_1;
- For each i, $1 \leq i \leq n-1$, π_{i+1} is a cut of a configuration in $C[\pi_1\pi_2\ldots\pi_i]$, and $C[\pi_1\pi_2\ldots\pi_{i+1}]$ is the set of line disjoint configurations obtained by cutting along π_{i+1} the configuration in $C[\pi_1\pi_2\ldots\pi_i]$ which contains π_{i+1}, and
- $C[\pi]$ is a set of line disjoint trivial configurations.

A simple induction shows that

Lemma 2.2 *The length of any cut sequence of C is equal to the number of modules in the layout corresponding to C minus 1.*

A cut sequence π of a configuration C is said to be *optimal* if it uses minimum number of L-cuts among all cut sequences of C. Figure 4 illustrates the cut sequence $\pi = L_1L_2V_1H_1V_4L_3V_2H_2H_5L_4H_4V_3H_3$ of a configuration with four L-cuts (L indicates L-cuts, H indicates horizontal S-cuts, and V indicates vertical S-cuts). The following observation relating a cut sequence of C to a feasible channel ordering of the layout corresponding to C was made in [9].

Theorem 2.3 *A sequence $\pi = \pi_1\pi_2\ldots\pi_n$ is a cut sequence of a configuration C if and only if $\pi^R = \pi_n\pi_{n-1}\ldots\pi_1$ defines a feasible channel ordering for the layout corresponding to C.*

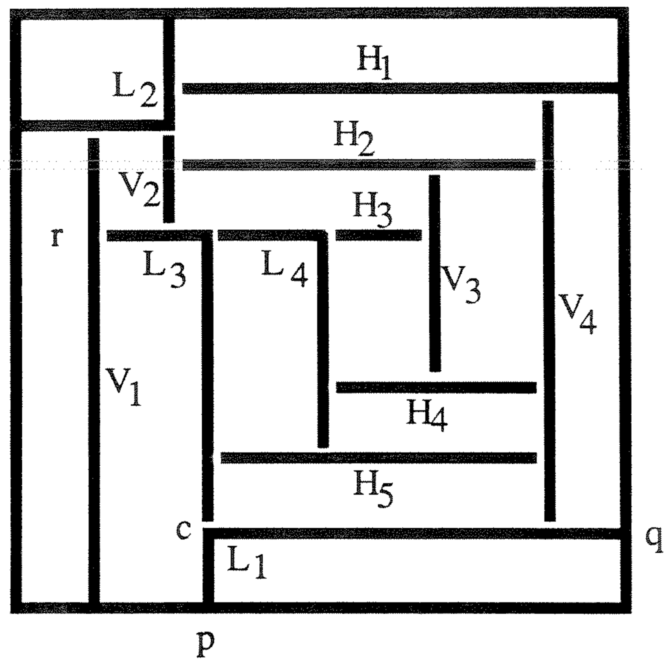


Figure 4: A cut sequence of a configuration

According to Theorem 2.3, the problem of computing a feasible channel ordering with minimum number of L-shaped channels for a layout of rectangular modules is equivalent to the problem of computing a cut sequence with minimum number of L-cuts for the corresponding configuration. The following corollary, which also appeared in [9], can be easily deduced from Lemma 2.1 and Theorem 2.3.

Corollary 2.4 *Every configuration has a cut sequence; Every layout of rectangular modules has a feasible channel ordering.*

A digraph $D = (V, A)$ is said to be *acyclic* if it does not contain a directed circuit, otherwise it is *cyclic*. Let $u, v \in V$ be vertices of D, then u is said to be a *predecessor* of v in D if there is a directed u-v path in D. A *strongly connected component* G of a digraph D is a maximal subgraph of D such that every pair of vertices lie in a common directed circuit of G. Let $S \subseteq V \times V - A$ be a set of arcs not in D, then an *acyclic augmentation of D with respect to S* is a subset $S^* \subseteq S$ such that $D^* = (V, A \cup S^*)$ is acyclic.

Given a cut sequence $\pi = \pi_1 \pi_2 \ldots \pi_n$ of a configuration C, we construct its *order constraint digraph* $D_\pi = (V_\pi, A_\pi)$ with vertex set $V_\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$. An arc (π_i, π_j) is in the arc set A_π if and only if one of the limits of π_j is on π_i. Figure 5 shows the order constraint

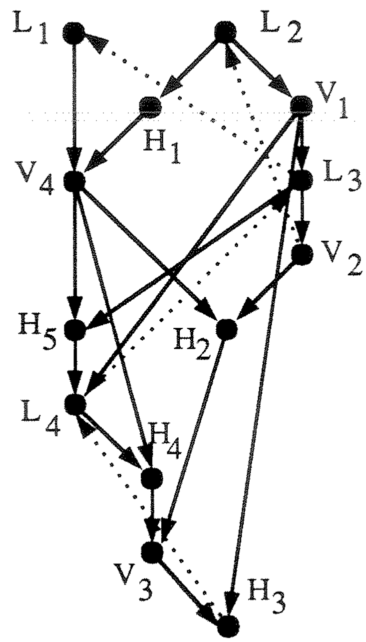


Figure 5: The order constraint digraph of a cut sequence

digraph of the cut sequence π illustrated in Figure 4 (only the solid arcs are in A_π). Note that $(V_1, H_3) \in A_\pi$ because the maximal (horizontal) line segment of C containing H_3 has an endpoint in V_1, *i.e.*, H_3 has a limit on V_1. Intuitively, if $(\pi_i, \pi_j) \in A_\pi$, then π_i has a straight line segment s which contains a limit p of π_j. Let t be the maximal line segment containing p and perpendicular to s, then all the cuts which contain part of t must be cut after the cut containing the part of s that contains p in any cut sequence of C. In particular, π_j must be cut after π_i in π, hence $i < j$.

We now state the following properties of order constraint digraphs.

Lemma 2.5 *The order constraint digraph D_π of a cut sequence is acyclic, planar, and has maximum in-degree ≤ 2.*

Proof: Let $\pi = \pi_1\pi_2 \ldots \pi_n$. As observed earlier, $(\pi_i, \pi_j) \in A_\pi$ implies $i < j$. Hence D_π is acyclic. To see that D_π is planar, we construct a planar drawing for it as follows: First, use any internal point of a cut to represent its corresponding vertex in D_π. For each arc $(\pi_i, \pi_j) \in A_\pi$, let s be the maximal line segment of C which has one endpoint on π_i and which contains a straight line segment of π_j. We represent the arc (π_i, π_j) by the shortest path of line segments of C between the points representing π_i and π_j which contains the part of s in π_j. The drawing so obtained has arc overlappings but no arc crossings. Arc overlappings can then be removed by replacing each line segment of C which is used by more than one arc by as many parallel line segments that run close to each other as arcs that use it. The drawing so obtained is a planar representation of D_π. Hence D_π is planar. Since each cut π has at most two limits, and each limit of a cut can induce at most one incoming arc, each vertex in D_π has in-degree at most two. □

Let α, β be cuts of a configuration C, α is said to be *dependent on* β if one straight line segment of α is completely contained in β. A cut of C is said to be *independent* if it is not dependent on any other cut of C, otherwise it is *dependent*. Since all S-cuts of C must be maximal line segments of C, we conclude that all S-cuts are independent. Note that if α is dependent on β, then after cutting β, one straight line segment of α is gone, and α becomes a straight line segment α' of a configuration in $C[\beta]$.

Given a configuration C, a *partition* of C is a set of line disjoint straight or L-shaped line segments of C whose union is the set of all internal line segments of C (*i.e.*, all the line segments of C except its boundaries). It is clear that the cuts of a cut sequence of C form a partition of C. A partition of C is said to be *admissible* if its line segments can be arranged into a cut sequence of C, *i.e.*, if it is the set of cuts of some cut sequence of C. We define the *order constraint digraph of a partition* of C similarly as the order constraint digraph of a cut sequence of C, and we have

Lemma 2.6 *If a partition of a configuration is admissible, then its order constraint digraph is acyclic.*

Note that the converse of Lemma 2.6 is not true in general. Figure 6 shows a partition of a configuration which is not admissible but has an acyclic order constraint digraph.

3 Previous Works

The idea of decomposing the routing area of a building-block layout of rectangular modules into straight and L-shaped channels was first introduced and shown to be always possible in [9]. The authors of [9] also proposed the following heuristic algorithm for computing a cut sequence of a configuration C which uses as few L-cuts as possible. (We use $x.y$ to denote concatenation of ordered sequences x and y. Also note that the algorithm is not presented in its original form, but rather it is presented in terms of our notation for consistency.)

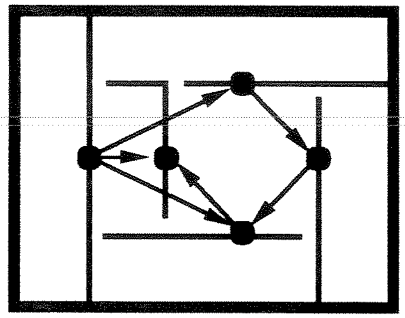


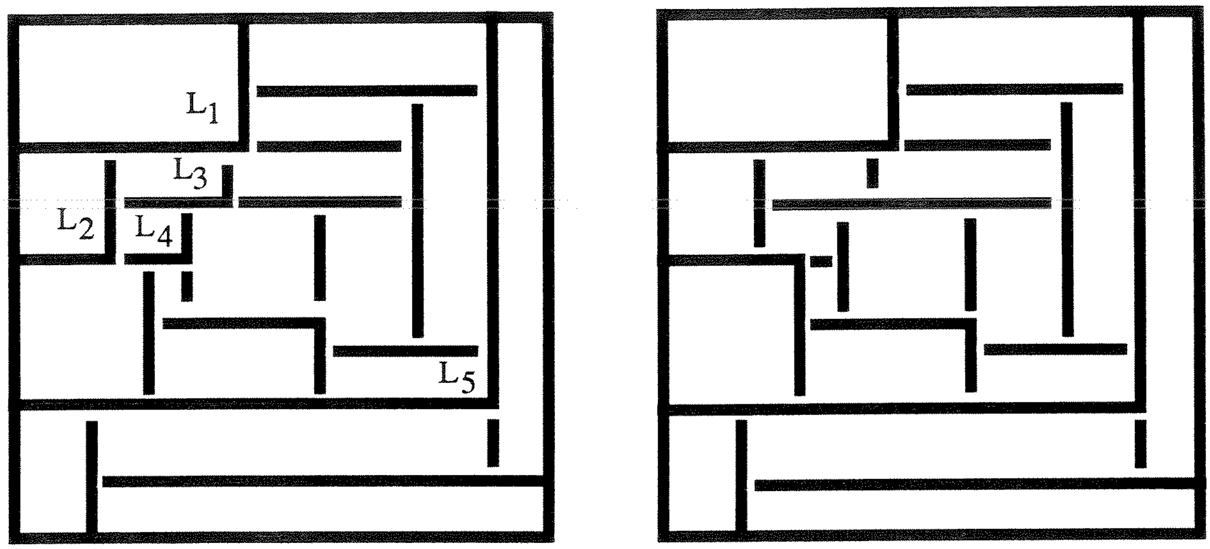
Figure 6: An inadmissible partition with acyclic order constraint digraph

Algorithm 3.1: Independence_Heuristic (C);
(* C is a configuration *)
Begin
$\pi := \epsilon$;
while $C[\pi]$ contains a nontrivial configuration **do**
if $C[\pi]$ contains a configuration which has an S-cut α
then $\pi := \pi.\alpha$;
else if $C[\pi]$ contains a configuration which has an independent L-cut β
then $\pi := \pi.\beta$;
else begin
Let γ be a cut of a configuration in $C[\pi]$;
$\pi := \pi.\gamma$
end;
return π
End.

By Lemma 2.1, at least one configuration of $C[\pi]$ has a cut if $C[\pi]$ is not a set of line disjoint trivial configurations. Hence Algorithm 3.1 works correctly and returns a cut sequence π of C. Referring to Figure 7(a), the configuration C has no S-cut and L_1 is the only independent L-cut of C. Hence Algorithm 3.1 cuts L_1. After that, it proceeds to cut L_2, L_3 and L_4 in that order. At this point, no configuration in $C[L_1L_2L_3L_4]$ has an S-cut or an independent L-cut, so the algorithm proceeds to cut any cut, say L_5, and so on. The cut sequence obtained in this way uses 6 L-cuts as shown in Figure 7(a). Note that there is a better cut sequence which uses only 4 L-cuts as shown in Figure 7(b). [3]

Algorithm 3.1 fails to be optimal because it makes decisions based only on local infor-

[3]It was claimed in [9] that Algorithm 3.1 is optimal if whenever the algorithm selects a cut, there is an S-cut or there is at most one independent cut The example in Figure 7 shows that this claim is not true in general.



(a) A cut sequence (b) A better cut sequence

Figure 7: Illustration of Algorithm 3.1

mations (it looks only at the cuts of a configuration) and makes no attempts of eliminating unnecessary L-cuts. In fact, it can produce poor results for large configurations. In the next section, we will show how can "mistakes" made in the early stage of the algorithm be "corrected" by maintaining some necessary global information.

4 Elimination of L-cuts

In this section, we describe a method of constructing a new cut sequence of a configuration C from a given cut sequence of C by changing some L-cuts into S-cuts and appropriately reordering the cuts. Thus the new cut sequence uses fewer L-cuts than the original one.

Given a cut sequence $\pi = \pi_1\pi_2\ldots\pi_n$ of C, we construct a set of arcs $R_\pi \subseteq V_\pi \times V_\pi$. An arc $(\pi_i, \pi_j) \in R_\pi$ if and only if π_j is an L-cut of a configuration in $C[\pi_1\pi_2\ldots\pi_{j-1}]$, and the corner of π_j is an endpoint of π_i. Each arc $a \in R_\pi$ is called a *recovery arc* of π. For the cut sequence π indicated in Figure 4, the set of recovery arcs is $R_\pi = \{(L_3, L_1), (V_2, L_2), (L_4, L_3), (H_3, L_4)\}$. They are indicated in Figure 4 by dotted arcs. Note that if $a = (\pi_i, \pi_j)$ is a recovery arc of π, then π_i cannot be a cut until after π_j is cut. Hence we must have $i > j$. The cut π_i is called the *tail* of a, and the L-cut π_j is called the *head* of a, it will also be referred to as the *L-cut of π corresponding to a*. We can similarly define a *recovery arc of a partition* of a configuration.

Consider the recovery arc (L_3, L_1). Adding it to D_π will not result in a cyclic digraph because there is not directed L_1-L_3 path in D_π. Hence it is possible to extend the vertical line segment of L_3 to include the vertical line segment of L_1 and cut L_3 before L_1. In this example, by appropriately reorder the cuts, we can obtain the following cut sequence $\pi' = L_2V_1H_1L_3H_6V_4V_2H_2H_5L_4H_4V_3H_3$. Notice that the L-cut L_1 in π becomes an S-cut H_6 in π', and hence we save one L-cut. In this case, we say L_1 is *eliminated* by L_3. In general, one can try to eliminate an L-cut corresponding to a recovery arc by executing the following function.

Function 4.1: Shuffle (π, a);

```
(* π = π_1 π_2 ... π_n is a cut sequence of a configuration C, a = (π_i, π_j) ∈ R_π *)
(* If successful, Shuffle(π, a) is a new cut sequence of C with one fewer L-cut than π *)
Begin
  Let s be the straight line segment of π_j which is colinear with a straight line segment of π_i;
  π_i^* := π_i ∪ s;          (* extend one straight line segment of π_i *)
  π_j^* := π_j − s;          (* change π_j into an S-cut *)
  π' := π_1 π_2 ... π_{j−1};
  ω := π_j^*;
  for k := j + 1 to i − 1 do
    if π_k is a cut of C[π']
      then π' := π'.π_k
      else ω := ω.π_k;
  return π'.π_i^*.ω.π_{i+1} π_{i+1} ... π_n
End.
```

Assume $\pi' = \text{Shuffle}(\pi, a)$ is a cut sequence of C, where $a = (\pi_i, \pi_j)$, then the set of cuts of π is the same as the set of cuts of π' except that π_i, π_j are replaced by π_i^* and π_j^*, respectively. The type of π_i (being an S-cut or an L-cut) is the same as π_i^*. However, π_j changes from an L-cut in π to an S-cut π_j^* in π'. Hence the number of L-cuts of π' is one fewer than the number of L-cuts of π.

Let Π be an admissible partition of a configuration C, and let $a = (\pi_i, \pi_j)$ be a recovery arc of Π. Let s be the straight line segment of π_j which is colinear with a straight line segment of π_i, and let $\pi_i^* = \pi_i \cup s$, $\pi_j^* = \pi_j - s$. Then a is said to be a *valid recovery arc of the partition* Π if

$$\Pi(a) = (\Pi - \{\pi_i, \pi_j\}) \cup \{\pi_i^*, \pi_j^*\}$$

is an admissible partition of C, *i.e.*, π_j can be eliminated.

A *valid recovery arc of a cut sequence* π of a configuration C is a valid recovery arc of the admissible partition of C formed by the cuts of π. Let $a = (\pi_i, \pi_j)$ be a recovery arc of π, then $\pi' = \text{Shuffle}(\pi, a)$ is a cut sequence of C if and only if every line segment of it is a cut of a configuration in $C[\pi']$ when it is appended to the end of π' in the function Shuffle(π, a). This holds for those line segments precede π_i^* in π' by construction. It also holds for those line segments succeed π_i^* in π', because all the line segments that precede them in π also precede them in π'. Hence π' is a cut sequence of C if and only if π_i^* is a cut of a configuration in $C[\pi']$ when it is appended to the end of π' in the function Shuffle(π, a). Define a *violating sequence of* $a = (\pi_i, \pi_j)$ *in* π as a sequence

$$\pi_i = \pi_{i_0}, \pi_{i_1}, \ldots, \pi_{i_m}, \pi_{i_{m+1}} = \pi_j$$

of cuts of π, such that $m \geq 1$, and for $0 \leq h \leq m$, either $(\pi_{i_h}, \pi_{i_{h+1}}) \in R_\pi$ or $(\pi_{i_{h+1}}, \pi_{i_h}) \in A_\pi$. Note that we have $i = i_0 > i_1 > \ldots > i_m > i_{m+1} = j$. An *end* of a line segment of C is either an endpoint or a limit of the line segment. We now have

Theorem 4.1 *Let $a = (\pi_i, \pi_j)$ be a recovery arc of a cut sequence π of a configuration C, then the following statements are equivalent:*

1. *a is a valid recovery arc of π;*
2. *$\pi' = \text{Shuffle}(\pi, a)$ is a cut sequence of C;*
3. *There is no violating sequence of a in π.*

Proof: Let Π, Π' be, respectively, the partition of C formed by the set of line segments of π and π'. Then Π is admissible. If π' is a cut sequence of C, then Π' is also admissible and hence a is a valid recovery arc of π by definition. Hence statement 2 implies statement 1. Suppose π' is not a cut sequence of C, then π_i^* is not a cut of a configuration in $C[\pi']$ when it is appended to the end of π' in the function Shuffle(π, a). Hence one of its endpoints p is not on the boundary of a configuration of $C[\pi']$ when it is appended to the end of π' in the function Shuffle(π, a). This endpoint must also be an endpoint of π_i, for the other endpoint of π_i^* is an endpoint of π_j, which is on the boundary of a configuration of $C[\pi']$ when π_i^* is appended to the end of π' in the function Shuffle(π, a). This is due to the fact that the other cut π_{i_1} of π that contains p was put into the sequence ω in the function Shuffle(π, a). If p is the corner of π_{i_1}, then $(\pi_i, \pi_{i_1}) \in R_\pi$, otherwise $(\pi_{i_1}, \pi_i) \in A_\pi$. Obviously $\pi_{i_1} \neq \pi_j$ because p is not the corner of π_j (otherwise it would not be an endpoint of π_i^* as well). Since π_{i_1} is put into ω in the function Shuffle(π, a) because it is not a cut of a configuration in $C[\pi']$ when the function Shuffle(π, a) tries to append to the end of π', an endpoint of it must be contained in a cut π_{i_2} of π which is also put into ω in the function Shuffle(π, a). If this endpoint of π_{i_1} is the corner of π_{i_2}, then $(\pi_{i_1}, \pi_{i_2}) \in R_\pi$, otherwise $(\pi_{i_2}, \pi_{i_1}) \in A_\pi$. If $\pi_{i_2} = \pi_j$, we have found a violating sequence of a in π. Otherwise, we claim that proceeding this way we must reach π_j and hence detect a violating sequence of a in π after a finite number of steps. This is because the number of cuts of π' is finite, if we cannot reach π_j in finite steps, then we must encounter some cut π_{i_k} of π twice in this process, *i.e.*, we have found a sequence

$$\pi_{i_k}, \pi_{i_{k+1}}, \ldots, \pi_{i_r}, \pi_{i_{r+1}} = \pi_{i_k}$$

, such that for $k \leq h \leq r$, either $(\pi_{i_h}, \pi_{i_{h+1}}) \in R_\pi$ or $(\pi_{i_{h+1}}, \pi_{i_h}) \in A_\pi$. Hence $i_k > i_{k+1} > \ldots > i_r > i_{r+1} = i_k$, a clear contradiction. Therefore, statement 3 implies statement 2. Finally, assume there is a violating sequence

$$\pi_i = \pi_{i_0}, \pi_{i_1}, \ldots, \pi_{i_m}, \pi_{i_{m+1}} = \pi_j$$

of a in π, then we claim that there exists a sequence

$$\pi_i^* = \pi'_{i_0}, \pi'_{i_1}, \ldots, \pi'_{i_{m'}}, \pi'_{i_{m'+1}} = \pi_i^*,$$

such that for $0 \leq k \leq m'$, $\pi'_{i_k} \in \Pi'$ and $\pi'_{i_{k+1}}$ has an end on π'_{i_k}. From which we can deduce that for $0 \leq k \leq m'$, π'_{i_k} must precede $\pi'_{i_{k+1}}$ in any ordering of the line segments of Π' which results in a cut sequence of C. However, this is impossible because $\pi'_0 = \pi'_{m'+1} = \pi_i^*$. Thus statement 1 implies statement 3. According to the definition of violating sequence, for $0 \leq k \leq m$, π_{i_k} has an end on $\pi_{i_{k+1}}$. In particular, $\pi_i = \pi_{i_0}$ has an end p on π_{i_1}. This end p of π_i must also be an end of π_i^*, for otherwise it is the corner of π_j which cannot be on $\pi_{i_1} \neq \pi_j$. Therefore, π_i^* has an end on π_{i_1}. Furthermore, π_{i_m} has an end q on $\pi_{i_{m+1}} = \pi_j$, hence on $\pi_i^* \cup \pi_j^*$. If q is on π_i^*, then we can take

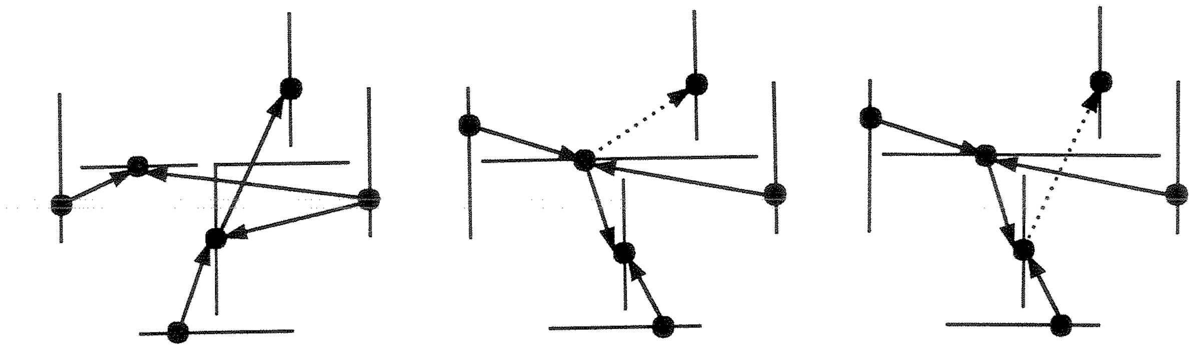
$$\pi_i^* = \pi_{i_0}, \pi_{i_1}, \ldots, \pi_{i_m}, \pi_{i_{m+1}} = \pi_i^*$$

as the sequence we are looking for, otherwise q is on π_j^* and we can take

$$\pi_i^* = \pi_{i_0}, \pi_{i_1}, \ldots, \pi_{i_m}, \pi_{i_{m+1}} = \pi_j^*, \pi_{i_{m+2}} = \pi_i^*$$

as the sequence we are looking for because π_j^* has an end on π_i^*. This completes the proof of the theorem. □

Using a procedure similar to Function 4.1, a violating sequence of a recovery arc of a cut sequence π of a configuration can be detected in $O(n)$ time, where n is the number of



(a) D_π (b) $D_{\pi'}$ (c) D'_π

Figure 8: Illustration of the proof of Lemma 4.2

modules in the layout corresponding to the configuration. Hence, according to Theorem 4.1, we can determine in linear time whether a recovery arc of π is a valid recovery arc of π.

Function 4.1 can be used to eliminate a single L-cut. In general, we would like to eliminate more than one L-cut at the same time. Let $a = (\pi_i, \pi_j)$ and $b = (\pi_k, \pi_l)$ be two recovery arcs of a cut sequence π of a configuration C, we say a and b are *independent* if either $i < l$ or $j > k$. (Note that $j < i$ and $l < k$.) Let P_π be the set of valid recovery arcs of π. An *independent acyclic augmentation of* D_π *with respect to* P_π is an acyclic augmentation of D_π with respect to P_π such that the recovery arcs in it are pairwise independent. Let R be a set of recovery arcs of a partition Π of a configuration C, and let $a = (\pi_i, \pi_j) \in R$, we define $\Pi(R) = \Pi'(R - \{a\})$ where $\Pi' = \Pi(a)$ and $\Pi(R) = \Pi$ if $R = \phi$. In the sequel, we will show that if P^* is an independent acyclic augmentation of D_π with respect to P_π, then $\Pi(P^*)$ is an admissible partition of C, where Π denotes the partition of C formed by the set of cuts of π, *i.e.*, there exists a cut sequence of C with $|P^*|$ fewer L-cuts than π.[4]

Lemma 4.2 *Let $a = (\pi_i, \pi_j)$ be a valid recovery arc of a cut sequence π of a configuration C, and let $\pi' = Shuffle(\pi, a)$. Also, let $D'_\pi = (V'_\pi, A'_\pi)$ be obtained from the digraph $D_\pi(a) = (V_\pi, A_\pi \cup \{a\})$ by renaming π_i, π_j to π^*_i, π^*_j, respectively. If for $u, v \in V_{\pi'} = V'_\pi$, there is a directed u-v path in $D_{\pi'}$, the order constraint digraph of the partition of C formed by the line segments of π', there is a directed u-v path in D'_π.*

Proof: The only arcs that are in $D_{\pi'}$ but not in D'_π are of the form (π^*_i, π_h) (the dotted arc in

[4]In the preliminary version of this paper [6], we claimed that if R^* is an acyclic augmentation of D_π with respect to R_π, the set of recovery arcs of π, then $\Pi(R^*)$ is an admissible partition of C. This is, however, not valid in general.

Figure 8(b)), where $(\pi_j, \pi_h) \in A_\pi$. According to the definition of D'_π, we have $(\pi^*_j, \pi_h) \in A'_\pi$ and $(\pi^*_i, \pi^*_j) \in A'_\pi$ because $a = (\pi_i, \pi_j)$ is an arc of $D_\pi(a)$. Hence there is a directed π^*_i-π_h path in D'_π. □

Theorem 4.3 *Let P^* be an independent acyclic augmentation of the order constraint digraph $D_\pi = (V_\pi, A_\pi)$ of a cut sequence π of a configuration C with respect to the set of valid recovery arcs P_π of π. Then, for any $a = (\pi_i, \pi_j) \in P^*$, $P^* - \{a\}$ is an independent acyclic augmentation of $D_{\pi'} = (V_{\pi'}, A_{\pi'})$ with respect to $P_{\pi'}$, where $\pi' = Shuffle(\pi, a)$.*

Proof: By Theorem 4.1, π' is a cut sequence of C. For $1 \leq h < j$ and $i < h \leq n$, π_h is identical to π'_h according to the Function 4.1. Because P^* is an independent acyclic augmentation of D_π with respect to P_π, for any $b = (\pi_k, \pi_l) \in P^* - \{a\}$, we have either $l < k < j$ or $i < l < k$. Hence b is a recovery arc of π' because its head $\pi_l = \pi'_l$ is an L-cut of π'. Furthermore, any violating sequence of b in π' is also a violating sequence of b in π. This is because any cut of π' in a violating sequence of b in π' must have index $\leq k$ and $\geq l$ in π', hence it is also a cut of π with the same index. Therefore, by Theorem 4.1, b must also be a valid recovery arc of π' because it is a valid recovery arc of π. Also, the recovery arcs in $P^* - \{a\}$ are pairwise independent, because the indices of their heads and tails are the same in π' as in π. It remains to show that $P^* - \{a\}$ is an acyclic augmentation of $D_{\pi'}$ with respect to $P_{\pi'}$. Obviously, $P^* - \{a\}$ is an acyclic augmentation of $D_\pi(a) = (V_\pi, A_\pi \cup \{a\})$ with respect to $P_\pi - \{a\}$, hence it is an acyclic augmentation of $D'_\pi = (V'_\pi, A'_\pi)$ with respect to $P_{\pi'}$ because $P^* - \{a\} \subseteq P_{\pi'}$. Hence by Lemma 4.2, it is also an acyclic augmentation of $D_{\pi'}$ with respect to $P_{\pi'}$. □

Given an independent acyclic augmentation P^* of the order constraint digraph of a cut sequence π of a configuration C with respect to the set of valid recovery arcs of π, Theorem 4.3 suggests a way to construct a new cut sequence of C from π with $|P^*|$ fewer L-cuts than π by repeatedly applying Function 4.1. However, this would require $O(n^2)$ time because each call to Function 4.1 require $O(n)$ time, where n is the length of π. The following function accomplishes the same thing in $O(n)$ time.

Function 4.2: Modify (π, P^*);

```
(* π = π1π2...πn is a cut sequence of a configuration C *)
(* P* is an independent acyclic augmentation of Dπ with respect to Pπ,
   the set of valid recovery arcs of π *)
Begin
  Sort the arcs in P* into ascending order of the indices of their heads;
  π' := ε;
  h := 0;
  while P* ≠ φ do
    begin
      Select the first arc a = (πi, πj) from P*;
      Let s be the straight line segment of πj which is colinear with a straight line segment of πi;
      πi* := πi ∪ s;
      πj* := πj − s;
      π' := π'.πh+1πh+2...πj−1;
      ω := πj*;
      for k := j + 1 to i − 1 do
        if πk is a cut of C[π']
          then π' := π'.πk
          else ω := ω.πk;
      π' := π'.πi*.ω;
      h := i;
      ω := ε;
      P* := P* − {a}
    end;
  return π'.πh+1πh+2...πn
End.
```

The correctness of Function 4.2 follows from the observation that because P^* is an independent acyclic augmentation of D_π with respect to P_π, the recovery arcs of P^* are pairwise noninterfering, *i.e.*, if $(\pi_i, \pi_j), (\pi_k, \pi_l) \in P^*$, $i \neq k$, then the intervals $[j, i]$ and $[l, k]$ are disjoint. In fact, if $j < l$, then $j < i < l < k$. Therefore, we have

Theorem 4.4 *Let P^* be an independent acyclic augmentation of the order constraint digraph of a cut sequence π of a configuration C with respect to the set of valid recovery arcs of π, then Modify(π, P^*) returns a cut sequence of C with $|P^*|$ fewer L-cuts in $O(n)$ time, where n is the length of π.*

Proof: Correctness follows from the above discussion. The arcs of P^* can be sorted in $O(n)$ time using bucket sort [1]. The while loop also terminates in $O(n)$ time because each cut of π is scanned for at most once within the loop. Hence the overall time complexity of Function 4.2 is $O(n)$. □

A maximum independent acyclic augmentation of D_π with respect to P_π, the set of valid recovery arcs of π, can be computed as follows:

Procedure 4.3: Maximum_Independent_Acyclic_Augmentation (π);
(* $\pi = \pi_1\pi_2 \ldots \pi_n$ is a cut sequence of a configuration C *)
Begin
Construct the order constraint digarph $D_\pi = (V_\pi, A_\pi)$ of π;
Compute the set of valid recovery arcs P_π of π;
$P^* := \phi$;
while $P_\pi \neq \phi$ **do**
begin
Select an arc $a = (\pi_i, \pi_j)$ from P_π such that i is minimal;
$P^* := P^* \cup \{a\}$;
Remove all arcs (π_k, π_l) such that $k \geq j$ from P_π
end;
return P^*
End.

Theorem 4.5 *Procedure 4.3 correctly computes a maximum independent acyclic augmentation of the order constraint digraph D_π of a cut sequence π of a configuration C with respect to the set of valid recovery arcs P_π of π in $O(n^2)$ time, where n is the length of π.*

Proof: It is clear that the valid recovery arcs returned by Procedure 4.3 are pairwise independent. Suppose $D_\pi^* = (V_\pi, A_\pi \cup P^*)$ is cyclic, and let $P = \pi_{i_0}, \pi_{i_1}, \ldots, \pi_{i_m}, \pi_{i_{m+1}}$ is a directed circuit of D_π^*, where $\pi_{i_{m+1}} = \pi_{i_0}$. Since each arc in P^* is a valid recovery arc of π, P must contain at least two arcs from P^*. Let $(\pi_{i_p}, \pi_{i_{p+1}}), (\pi_{i_q}, \pi_{i_{q+1}}) \in P^*$ be two consecutive recovery arcs of P *i.e.*, there is no recovery arc between them in P) with $p < q$, then we have $i_{p+1} < i_{p+2} < \ldots < i_q$. Hence we also have $i_p < i_{q+1}$ because the recovery arcs of P^* are pairwise independent. Let $(\pi_{i_u}, \pi_{i_{u+1}})$ and $(\pi_{i_v}, \pi_{i_{v+1}})$ be the first and last recovery arcs of P, by applying the above argument to each consecutive pair of recovery arcs of P we can conclude that $i_u < i_{v+1}$. On the other hand, we have $i_{v+1} < i_{v+2} < \ldots < i_m < i_{m+1} = i_0 < i_1 < \ldots < i_u$, a contradiction. Hence P^* is an independent acyclic augmentation of D_π. Finally, assume P^* is not a maximum independent acyclic augmentation of D_π, then there exists a maximum independent acyclic augmentation P' of D_π which shares the maximum number of common arcs with P^*. Let a be the arc of P^* with minimum head index which is not in P'. According to Procedure 4.3, a is the arc with minimum head index among all arcs that are pairwise independent with the arcs in P^* (P') whose head index is smaller than the head index of a. Hence P' has at most one arc b such that a and b are not independent. Therefore, $P'' = (P' - \{b\}) \cup \{a\}$ is a maximum independent acyclic augmentation of D_π

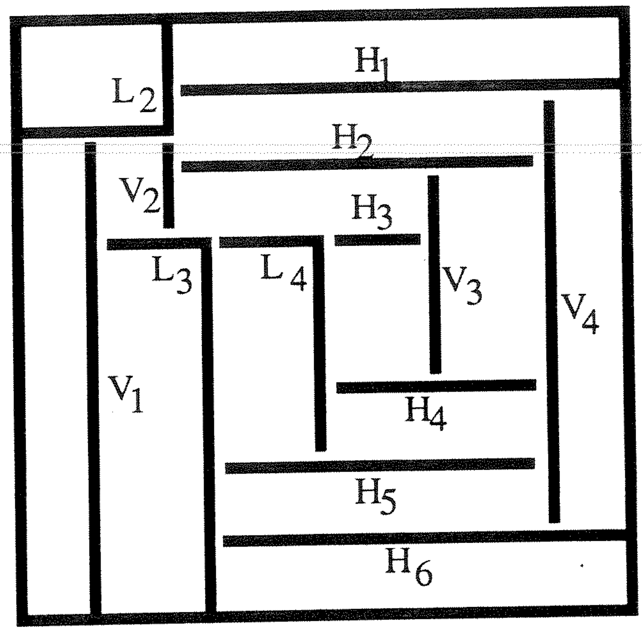


Figure 9: Another cut sequence

having more common arcs with P^* than P' does, contradicting the choice of P'. This completes the proof of the theorem. □

Based on Theorem 4.4, Algorithm 3.1 can be modified as follows:

Algorithm 4.4: Modified_Independence_Heuristic (C);
(* C is a configuration *)
Begin
Call Algorithm 3.1 to compute a cut sequence π of C;
Construct the order constraint digraph $D_\pi = (V_\pi, A_\pi)$ of π;
Compute the set of valid recovery arcs P_π of π;
Call Procedure 4.3 to compute a maximum independent acyclic augmentation P^* of D_π with respect to P_π;
return Modify(π, P^*)
End.

The worst case time complexity of Algorithm 4.4 is $O(n^2)$, where n is the number of modules in the layout corresponding to the given configuration. Algorithm 4.4 always produces a cut sequence which uses no more L-cuts than the result produced by Algorithm 3.1. For example, applying Algorithm 4.4 to the configuration in Figure 4 we obtain the cut sequence indicated in Figure 9 which uses 3 L-cuts, one fewer than the cut sequence produced by Algorithm 3.1. However, the result produced by Algorithm 4.4 is dependent on the result produced by Algorithm 3.1, which is used as the initial cut sequence in Algorithm 4.4. Hence it is very important to construct a "good" initial cut sequence. This is the topic to be discussed in the next section.

5 Construction of Initial Cut Sequences

The basic idea of our algorithm is to start out with an initial cut sequence, and then to transform it into a better cut sequence by eliminating as many L-cuts as possible. Therefore, the final result produced by our algorithm depends both on the initial cut sequence and the number of L-cuts eliminated. Let π, π' be two cut sequence of a configuration C, then π' is said to be *derivable* from π if $\Pi' = \Pi(R)$ for some set of recovery arcs R of π, where Π, Π' are, respectively, the partition of C formed by the cuts of π and π'. The way we construct our initial cut sequence is motivated by the following results.

Theorem 5.1 *Let π and π' be two cut sequences of a configuration C with no "+" type junctions.* [5] *If π' contains all the L-cuts of π, then there exists a set of recovery arcs R' of π', such that $\Pi'(R')$ is admissible and all the L-shaped line segments of $\Pi'(R')$ are L-cuts of π, where Π' is the partition of C formed by the cuts of π',* i.e., *there exists a cut sequence of C derivable from π' whose L-cuts are all L-cuts of π.*

Proof: By induction on N, the length of π (π'). The theorem holds vacuously for $N = 0$. Assume $N > 0$, and consider the first cut α of π which separates C into two line disjoint subconfigurations C_1 and C_2. We consider the following two cases:

- Case 1: α is also a cut of π'.
  Let $\pi_\alpha(i)$, $\pi'_\alpha(i)$, $i = 1, 2$, be, respectively, the subsequence of π, π' consisting of all the cuts of π, π' which are line segments of C_i, then $\pi_\alpha(i)$, $\pi'_\alpha(i)$ are cut sequences of C_i, $i = 1, 2$. Furthermore, $|\pi_\alpha(1)| = |\pi'_\alpha(1)| < |\pi| = N$, $|\pi_\alpha(2)| = |\pi'_\alpha(2)| < N$, and all L-cuts of $\pi_\alpha(i)$ are L-cuts of $\pi'_\alpha(i)$, $i = 1, 2$. Hence by the inductive hypothesis, for $i = 1, 2$, there exists a recovery arc set R'_i of $\pi'_\alpha(i)$ such that $\Pi'_i(R'_i)$ is admissible and all the L-shaped line segments of $\Pi'_i(R'_i)$ are L-cuts of $\pi_\alpha(i)$, where Π'_i is the partition of C_i formed by the cuts of $\pi'_\alpha(i)$. Therefore, $R' = R'_1 \cup R'_2$ is a recovery arc set of π' satisfying the conditions listed in the theorem.

[5] By slightly generalizing the notation of recovery arcs so that the way of decomposing a "+" type junction can be affected by a recovery arc (by letting S-cuts recovering S-cuts), the restriction of C having no "+" type junction can be released.

- Case 2: α is not a cut of π'.

  In this case, α must be an S-cut. Since C does not contain any "+" type junctions, α contains at most one S-cut of π', and there is a natural one-to-one correspondence between the set of recovery arcs of a cut sequence of C and the set of L-cuts of that cut sequence. Because all L-cuts of π are also L-cuts of π', the set of L-cuts of π' having a line segment contained in α contains no L-cuts of π. Let R_α be the set of recovery arcs of π' corresponding to this set of L-cuts of π'. It is easy to see that $\Pi'(R_\alpha)$ is admissible and there is a way of arranging the line segments of $\Pi'(R_\alpha)$ into a cut sequence π'' of C such that α is the first cut of π''. Since π'' contains all the L-cuts of π, according to Case 1, there exists a set of recovery arcs R''_α of π'' such that $\Pi''(R''_\alpha)$ is admissible and all the L-shaped line segments of $\Pi''(R''_\alpha)$ are L-cuts of π, where Π'' is the partition of C formed by the cuts of π''. Hence $R' = R_\alpha \cup R''$ is the set of recovery arcs of π' satisfying the conditions listed in the theorem, where R'' is the set of recovery arcs of π' corresponding to the set of recovery arcs R''_α of π''.

Hence the theorem follows. □

Corollary 5.2 *Let π be a cut sequence of a configuration C with no "+" type junctions containing all the L-cuts of an optimal cut sequence π' of C, then π' is derivable from π.*

Since the "+" type junctions of a configuration can be split into two "L" type junctions [3, 8], according to Corollary 5.2, we would like to construct our initial cut sequence for a configuration C such that it contains all the L-cuts of an optimal cut sequence of C. However, we do not know of such a set of L-cuts. Therefore, what we will do in the sequel is to study the properties of the cuts which are members of some optimal cut sequence of C, and we will try to construct our initial cut sequence in such a way that at least one cut with such properties is retained in each stage of the algorithm.

Lemma 5.3 *Let α be a cut of a configuration C cutting along which separating C into two disjoint configurations C_1 and C_2, and let m, m_1 and m_2 be the number of L-cuts in an optimal cut sequence of C, C_1 and C_2, respectively. Then $m_1 + m_2 \leq m \leq m_1 + m_2 + \delta_\alpha$, where $\delta_\alpha = 1$ if α is an L-cut, otherwise $\delta_\alpha = 0$.*

Proof: It is clear that $m \leq m_1 + m_2 + \delta_\alpha$. To show that $m \geq m_1 + m_2$, observe that the cuts of an optimal cut sequence π of C can be divided into the following three types:

1. Cuts that are completely contained in either C_1 or C_2;

2. Cuts that have part of them in C_1 and part of them in C_2; and

3. Cuts that contain part of α.

We can construct from π a cut sequence $\pi(1)$ of C_1, whose cuts are:

1. The cuts of π that are contained in C_1; and

2. The parts of the cuts of π of types (2) and (3) in C_1.

Similarly, we can construct a cut sequence $\pi(2)$ of C_2 from π. Each L-cut of π of type (1) is transformed into an L-cut of either $\pi(1)$ or $\pi(2)$; Each L-cuts of π of type (2) is transformed into an S-cut of $\pi(1)$ or $\pi(2)$ and an L-cut of $\pi(2)$ or $\pi(1)$; Each L-cut of π of type (3) is transformed into an S-cut of either $\pi(1)$ or $\pi(2)$. Therefore, the number of L-cuts in π, which is equal to m, is greater than or equal to the sum of the number of L-cuts in $\pi(1)$ and $\pi(2)$, which is in turn greater than or equal to $m_1 + m_2$. □

If $m = m_1 + m_2 + \delta_\alpha$ in Lemma 5.3, then α is said to be an *essential cut* of C. Hence all S-cuts are essential. Any configuration has an essential cut. For any essential cut α of a configuration C, there exists an optimal cut sequence of C with α as its first cut. Hence essential cuts are never "wrong" cuts to cut. The following theorem characterizes essential cuts.

Theorem 5.4 *A cut sequence $\pi = \pi_1\pi_2\ldots\pi_n$ of a configuration C is an optimal cut sequence of C if and only if for $1 \le k \le n$, π_k is an essential cut of a configuration in $C[\pi_1\pi_2\ldots\pi_{k-1}]$.*

Proof: Let $\pi = \pi_1\pi_2\ldots\pi_n$ be a cut sequence of a configuration C, such that for $1 \le k \le n$, π_k is an essential cut of a configuration in $C[\pi_1\pi_2\ldots\pi_{k-1}]$. Assume, to the contrary, that π is not an optimal cut sequence of C, then there exists an optimal cut sequence $\pi' = \pi'_1\pi'_2\ldots\pi'_n$ of C, such that for $1 \le j \le i < n$, $\pi_j = \pi'_j$, and i is maximized, *i.e.*, π' is an optimal cut sequence of C having longest common prefix with π. Since π_{i+1} is an essential cut of a configuration in $C[\pi_1\pi_2\ldots\pi_i]$, there exists an optimal cut sequence of C of the form $\pi'' = \pi_1\pi_2\ldots\pi_i\pi_{i+1}\pi''_{i+2}\ldots\pi''_n$. However, this contradicts the choice of π' because π'' is an optimal cut sequence of C having longer common prefix with π than π'. This proves the sufficiency. To prove the necessity, assume $\pi = \pi_1\pi_2\ldots\pi_n$ is an optimal cut sequence of C and π_1 is not an essential cut of C (hence it is an L-cut). Let C be separated by π_1 into two line disjoint subconfigurations C_1 and C_2, and let m, m_1 and m_2 be the number of L-cuts of an optimal cut sequence of C, C_1 and C_2, respectively. Then $m = m_1 + m_2$ according to Lemma 5.3, and the number of L-cuts in π is at least $m_1 + m_2 + 1$, contradicting the assumption that π is an optimal cut sequence of C. Hence π_1 must be an essential cut of C. Similarly, we can show that for $1 \leq k \leq n$, π_k is an essential cut of a configuration in $C[\pi_1\pi_2 \ldots \pi_{k-1}]$. □

Since the corner of a definite L-cut of a configuration is an "L" type junction of the configuration, it must be contained in an L-cut in any cut sequence of the configuration. Since a definite L-cut is the "largest" L-cut that contains its corner, we have

Corollary 5.5 *Any definite L-cut of a configuration is also an essential cut of the configuration.*

A cut α of C is said to be *better* than another cut β of C if the minimal number of L-cuts in a cut sequence of C having α as its first cut is smaller than the minimal number of L-cuts in a cut sequence of C having β as its first cut. It is easy to see that no cut of C is better than an essential cut of C. We now have

Lemma 5.6 *Let α, β be cuts of a configuration C such that α is dependent on β, then α cannot be better than β.*

Proof: Since α is dependent on β, it is an L-cut of C. After cutting β, one straight line segment of α is gone and α becomes a straight line segment α' of a configuration in $C[\beta]$. Let $C[\alpha] = \{C_1, C_2\}$ and $C[\beta\alpha'] = \{C_1, C_3, C_4\}$ where $C_2[\alpha'] = \{C_3, C_4\}$. For $1 \leq i \leq 4$, let m_i be the number of L-cuts in an optimal cut sequence of C_i, and let m_α, m_β be, respectively, the minimum number of L-cuts in any cut sequence of C with α, β as its first cut. Then $m_2 \geq m_3 + m_4$ by Lemma 5.3. Hence

$$\begin{aligned} m_\beta &\leq \delta_\beta + m_1 + m_3 + m_4 \\ &\leq m_1 + m_2 + 1 \\ &\leq m_\alpha \end{aligned}$$

where $\delta_\beta = 1$ if β is an L-cut of C, otherwise $\delta_\beta = 0$. □

Corollary 5.7 *Let α, β be cuts of a configuration C. If there exists an order sequence $\alpha = c_0, c_1, \ldots, c_m, c_{m+1} = \beta$ of cuts of C such that for $0 \leq k \leq m$, c_k is dependent on c_{k+1}, then α cannot be better than β.*

In general, given a configuration C, we will not always be able to find an essential of C. However, we can identify a set of cuts such that one of them must be an essential cut of C.

For a given configuration C, we construct its *cut dependence digraph* $D_C = (V_C, A_C)$ having the cuts of C as it vertices. Two cuts α and β of C are joined by an arc $(\alpha, \beta) \in A_C$ if and only if α is dependent on β. A cut α is said to be *dominated* be another cut β if there exists a directed α-β path in D_C, but there does not exist a directed β-α path in D_C. A cut of C is a *dominated cut* if it is dominated by some other cut of C.

The maximal (induced) subgraph $D'_C = (V'_C, A'_C)$ of the cut dependence digraph $D_C = (V_C, A_C)$ of a configuration C with the set of non-dominated cuts of C as its vertex set is called the *reduced cut dependence digraph* of C. Note that for any cut α of C, there exists a cut $\beta \in V'_C$ such that there exists a directed α-β path in D_C. A *set of representative cuts* of C consists of either an S-cut or a definite L-cut of C, if C has such a cut, or one cut from each strongly connected component of its reduced cut dependence digraph. The cardinality of a set of representative cuts of C is called the *characteristic* of C.

Lemma 5.8 *Each set of representative cuts of a configuration C contains at least one essential cut of C.*

Proof: Let Ω be a set of representative cuts of C. If Ω contains either an S-cut or a definite L-cut, then it contains an essential cut of C. Otherwise Ω contains one cut from each strongly connected component of D'_C, the reduced cut dependence digraph of C. Hence for any cut α of C, there exists a directed α-β path in D_C for some cut $\beta \in \Omega$. Therefore, α cannot be better than β by Corollary 5.7. In particular, if α is an essential cut of C, then β is also an essential cut of C by Theorem 5.4. Since C has an essential, Ω contains an essential cut of C. □

We can now present our algorithm for constructing an initial cut sequence. In Procedure 5.1 below, one or more cuts are cut at each stage, such that at least one of them is an essential cut.

Procedure 5.1: Initial_Cut_Sequence (C);
(* C is a configuration without "+" type junctions *)
Begin
$\pi := \epsilon$;
while $C[\pi]$ contains a nontrivial configuration **do**
begin
Select from $C[\pi]$ a nontrivial configuration X with minimal characteristic;
Compute a set of representative cuts $\Omega = \{\omega_1, \omega_2, \ldots, \omega_m\}$ of X;
$\pi := \pi.\omega_1\omega_2\ldots\omega_m$
end;
return π
End.

Theorem 5.9 *Procedure 5.1 correctly constructs a cut sequence of a configuration without "+" type junctions in* $O(n^2)$ *time, where* n *is the number of modules in the layout corresponding to the given configuration.*

Proof: Since C has no "+" type junctions, any two distinct cuts in a set of representative cuts of C are line disjoint and noninterfering, *i.e.*, in each iteration of the while loop in Procedure 5.1, ω_k is a cut of a configuration in $C[\pi.\omega_1\omega_2\ldots\omega_{k-1}]$ for $1 \leq k \leq m$. Hence Procedure 5.1 returns a cut sequence of C. Each iteration of the while loop in Procedure 5.1 can be implemented to run in $O(n)$ time, and the while loop is executed $n-1$ time. Therefore, the overall time complexity of Procedure 5.1 is $O(n^2)$. □

Theorem 5.10 *If* $|\Omega| = 1$ *in each iteration of the while loop in Procedure 5.1, then Procedure 5.1 produces an optimal cut sequence for* C.

Proof: In this case, the cut sequence of C returned by Procedure 5.1 satisfies the conditions listed in Theorem 5.4, hence it is optimal. □

6 A New Algorithm

We present in this section our main algorithm. It is a refinement of Algorithm 4.4. The algorithm first constructs an initial cut sequence using using Procedure 5.1. Next it uses Procedure 4.3 to compute a maximum independent acyclic augmentation P^* of the order constraint digraph D_π of π with respect to P_π, the set of valid recovery arcs of π. It then eliminates the set of L-cuts of π corresponding to P^* by using Function 4.2.

Algorithm 6.1: Essentiality_Heuristic (C);
(* C is a configuration *)
Begin
Split "+" type junctions of C into "L" type junctions;
Call Procedure 5.1 to construct an initial cut sequence π for C;
Construct the order constraint digraph $D_\pi = (V_\pi, A_\pi)$ of π;
Compute the set of valid recovery arcs P_π of π;
Call Procedure 4.3 to compute a maximum independent acyclic augmentation P^* of D_π with respect to P_π;
$\pi := \text{Modify}(\pi, P^*)$;
return π
End.

Theorem 6.1 *Algorithm 6.1 correctly computes a cut sequence for the input configuration C in $O(n^2)$ time, where n is the number of modules in the layout corresponding to C.*

Proof: The correctness of the algorithm follows from Theorems 5.9, 4.4 and 4.5. Each step of the algorithm takes at most $O(n^2)$ time. Hence the worst case time complexity of the whole algorithm is $O(n^2)$. □

7 Experimental Results

We have implemented Algorithm 6.1 in Pascal language on a SUN SPARCstation 1. We compared our results with the results produced by the algorithm in [9] (Algorithm 3.1). These results are summarized in Table 1. Here "size" refers to the number of modules in the layout corresponding to the given configuration. As we can see from the table, improvements of up to 29% were achieved. For all these examples, our algorithm took less than one tenth of a second of CPU time to finish the computation.

Figure 10 shows the result obtained by Algorithm 6.1 for a large configuration consisting of 136 channels. L-cuts are indicated by thick lines. Forty-five L-cuts were used, in contrast to the result produced by Algorithm 3.1 which uses 56 L-cuts.

8 Concluding Remarks

We present in this paper a fast algorithm for minimizing the number of L-shaped channels in decomposing the routing area of a build-block layout of rectangular modules into straight and L-shaped channels. Although the algorithm is not optimal, it consistently produces significantly better results than a previously known algorithm proposed in [9]. The algorithm

EX.	*Size*	*Alg. 3.1*	*Alg. 6.1*	*Improvement*
Ex. 1	14	4	3	25.0%
Ex. 2	63	31	24	22.6%
Ex. 3	30	12	10	16.7%
Ex. 4	36	14	10	28.6%
Ex. 5	46	19	15	21.1%
Ex. 6	136	56	45	19.6%
Ex. 7	82	34	28	17.6%
Ex. 8	58	23	19	17.4%

Table 1: Experimental results

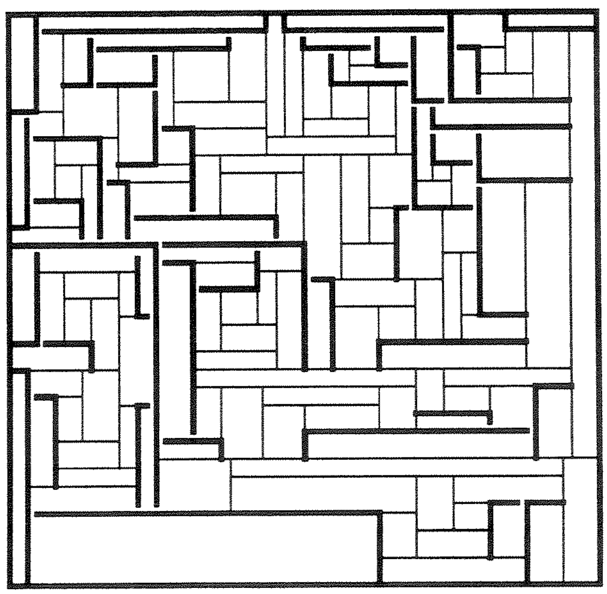


Figure 10: A large example

is very efficient. It took less than one tenth of a second of CPU time on a SUN SPARCstation 1 to finish the computation for examples with up to 136 channels. Significant improvements over the algorithm in [9] were observed.

Our algorithm uses the novel method of using global information to "correct" earlier decision errors made based on local information. The algorithm can be partitioned into two phases. In the first phase, a simple heuristic approach is used which makes decisions based only on local information, global information is maintained as the process proceeds. These global information are used in the second phase to "correct" mistakes made in the first phase. This method can conceivably produce better results than an algorithm using only the heuristic approach used in the first phase. Final results can be further improved by carefully constructing the initial solution. We believe that this strategy is applicable in many other situations as well.

The method we use is a transformational one. It starts out with an initial solution and transforms it into a better solution. It fails to be optimal because the transformation we use is not strong enough, *i.e.*, it only allows us to change L-cuts into S-cuts, but not the other way around (there is no uphill moves). This kind of transformation cannot guarantee to bring us from any solution to an optimal solution. It would be very interesting to see either a polynomial time optimal algorithm for solving this problem, or a proof of NP-hardness of the problem.

Throughout this paper, we have focused our attention on layouts consisting only of rectangular modules. For general layouts of arbitrary rectilinear modules, it is necessary to decompose the routing area into more complicated regions, such as staircase channels and nonstaircase channels [13] in order to have a feasible channel ordering, *i.e.*, an order of routing these regions so that whenever a region is routed, its dimensions can be adjusted without destroying the previously routed regions. Heuristic algorithms have been proposed for such general layouts [13]. It is interesting to see if our method can be applied in the more general setting to obtain algorithms which optimize some reasonably chosen metric.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The design and analysis of computer algorithms*, Addison Wesley, Reading, MA, 1974.

[2] M. Burstein and R. Pelavin, "Hierarchical channel router", *INTEGRATION, the VLSI journal, vol.* **1**, 21-38, 1983.

[3] H. Cai, "Connectivity based channel construction and ordering for building-block layout", *Proc. of the 25th Design Automation Conference*, 560-565, 1988.

[4] H. Cai, "On empty rooms in floorplan graphs: comments on a deficiency in two papers", *IEEE Trans. on CAD, CAD-***8**, 795-797, 1989.

[5] Y. Cai and D.F. Wong, "A channel/switchbox definition algorithm for building-block layout", *Proc. of the 27th ACM/IEEE Design Automation Conference*, 638-641, 1990.

[6] Y. Cai and D.F. Wong, "On minimizing the number of L-shaped channels", *Proc. of the 28th ACM/IEEE Design Automation Conference*, 328-334, 1991.

[7] H.H. Chen, "Routing L-shaped channels in nonslicing-structure placement", *Proc. of the 24th ACM/IEEE Design Automation Conference*, 152-158, 1987.

[8] T. Chiba, N. Okuda, T. Kambe, I. Nishioka and S. Kimura, "SHARPS: a hierarchical layout system for VLSI", *Proc. of the 18th ACM/IEEE Design Automation Conference*, 820-827, 1981.

[9] W.M. Dai, T. Asano and E. Kuh, "Routing region definition and ordering scheme for building-block layout", *IEEE Trans. on CAD, CAD-***4**, 189-197, 1985.

[10] D.N. Deutsch, "A dogleg channel router", *Proc. of the 13rd ACM/IEEE Design Automation Conference*, 425-433, 1976.

[11] C.M. Fiduccia and R.L. Rivest, "A greedy channel router", *Proc. of the 19th ACM/IEEE Design Automation Conference*, 418-424, 1982.

[12] M.R. Garey and D.S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, W.H. Freeman & Co., New York, NY, 1979.

[13] M. Guruswamy and D.F. Wong, "Channel routing order for building-block layout with rectilinear modules", *Proc. of the 1988 IEEE International Conference of Computer-Aided Design*, 184-187.

[14] A. Hashimoto and J. Stevens, "Wire routing by optimizing channel assignment within large apertures", *Proc. of the Eighth ACM/IEEE Design Automation Workshop*, 155-163, 1971.

[15] C.P. Hsu, "A new two-dimensional routing algorithm", *Proc. of the 19th ACM/IEEE Design Automation Conference*, 393-402, 1982.

[16] T.C. Hu and E.S. Kuh, ed., *VLSI circuit layout: theory and design*, IEEE Press, New York, NY, 1985.

[17] S. Kimura, N. Kubo, T. Chiba and I. Nishioka, "An automatic routing scheme for general cell LSI", *IEEE Trans. on CAD, CAD-***2**, 285-292, 1983.

[18] T. Ohtsuki, ed., *Advances in CAD for VLSI, volume 4: layout design and verification*, Amsterdam, The Netherlands, North Holland, 1986.

[19] R. H. Otten, "Automatic floorplan design", *Proc. of the 19th ACM/IEEE Design Automation Conference*, 261-267, 1982.

[20] B.T. Preas, "Placement and routing algorithms for hierarchical integrated circuit layout", *Ph.D. Dissertation*, Stanford Univ., 1979.

[21] B. Preas and M. Lorenzetti, ed., *Physical design automation of VLSI systems*, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, CA, 1988.

[22] B.T. Preas and W.M. vanCleemput, "Routing algorithm for hierarchical IC layout", *Proc. of the 1979 IEEE International Symposium on Circuits and Systems*, 482-485.

[23] J. Reed, A. Sangiovanni-Vincentelli, and M. Santomauro, "A new symbolic channel router: YACR2", *IEEE Trans. on CAD, vol. CAD-***4**, 208-219, 1985.

[24] J. Soukup, "Circuit layout", *Proc. IEEE vol.* **69**, 1281-1304, 1981.

[25] T. Yoshimura and E.S. Kuh, "Efficient algorithms for channel routing", *IEEE Trans. on CAD, vol. CAD-***1**, 25-35, 1982.