TWO GEOMETRIC OPTIMIZATION PROBLEMS

BHASKAR DASGUPTA
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455-0159
email: dasgupta@cs.umn.edu

and

VWANI ROYCHOWDHURY
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907-1285
email: vwani@drum.ecn.purdue.edu

Abstract. We consider two optimization problems with geometric structures. The first one concerns the following minimization problem, termed as the *rectilinear polygon cover* problem: "Cover certain features of a given rectilinear polygon (possibly with rectilinear holes) with the minimum number of rectangles included in the polygon." Depending upon whether one wants to cover the interior, boundary or corners of the polygon, the problem is termed as the *interior, boundary* or *corner* cover problem, respectively. Most of these problems are known to be NP-complete. In this chapter we survey some of the important previous results for these problems and provide a proof of impossibility of a polynomial-time approximation scheme for the interior and boundary cover problems. The second problem concerns routing in a segmented routing channel. The related problems are fundamental to routing and design automation for Field Programmable Gate Arrays (FPGAs), a new type of electrically programmable VLSI. In this chapter we survey the theoretical results on the combinatorial complexity and algorithm design for segmented channel routing. It is known that the segmented channel routing problem is in general NP-Complete. Efficient polynomial time algorithms for a number of important special cases are presented.

Key words: Rectilinear Polygons, Segmented Channel Routing, Approximation Heuristics

1. Introduction

The problem of covering a certain class of features of a rectilinear polygons with the minimum number of rectangles belongs to a more general class of geometric covering and decomposition problems. Depending upon whether one wants to cover the interior, boundary or corners of the polygon, such a problem is termed as *interior, boundary or corner* cover problem, respectively. The rectilinear cover problem has received particular attention, partly because very little progress has been made in finding efficient algorithms for covering arbitrary polygons with primitive shapes, and also partly because the rectilinear cover problem has important applications in storing images[19], and in the manufacture of integrated circuits[20]. Also, an investigation of this problem has given rise to special kinds of perfect graphs of interest[21]. Unfortunately, most of these problems are NP-complete in general, hence there is need to develop efficient heuristics for these problems. In this chapter we survey some previous results for these problems and provide proofs of some very recent results in this area.

Conventional channel routing [16] concerns the assignment of a set of connections to tracks within a rectangular region. The tracks are freely customized by the appropriate mask layers. Even though the channel routing problem is in general NP-Complete [26], efficient heuristic algorithms exist and are in common use in many placement and routing systems. In this chapter we study the more restricted channel routing problem (see Fig. 13), where the routing is constrained to use fixed wiring segments of predetermined lengths and positions within the channel. Such segmented channels are incorporated in channeled Field Programmable Gate Arrays (FPGAs) [12]. In [11, 25] it is demonstrated that a well designed segmented channel needs only a few tracks more than a freely customized channel. The problem of segmented channel routing is shown to be NP-Complete in [11, 25]. In this chapter we survey some of the polynomial time algorithms for cases with special geometrical structures.

The rest of the chapter is organized as follows:

– In Section 2.1 we state the basic definitions and provide a precise statement of the rectilinear polygon cover problem.
– In Section 2.2 we state results about the complexity of an exact solution for most of the above problems.
– In Section 2.3 we state various approximation heuristics for approximating these problems.
– In Section 2.4 we state some results which gives polynomial time solutions for some special cases of the rectilinear polygon cover problem.
– In Section 2.5 we prove a result showing the impossibility of having a polynomial-time approximation scheme for the interior and boundary cover problem under the assumption of P≠NP.
– In Sections 3.1 and 3.2 we provide a description of segmented channel routing, introduce the related optimization problems and survey some of the previous results.
– In Section 3.3 we introduce some of the polynomial time algorithms for segmented channel routing.
– We conclude in Section 4 with some open questions about these problems.

2. The Rectilinear Polygon Cover Problems

2.1. Definitions and Preliminaries

A *rectilinear polygon* P is a polygon with its sides parallel to the coordinate axes. Such a polygon may or may not have holes, but if the holes are present they are also rectilinear. The polygon P is said to be in *general position* if no three of its vertices are on the same horizontal or vertical line. In all subsequent discussions we assume that the given polygon is *simple*, *i. e.,* no two non-consecutive edges of the polygon cross each other.

The corners of the given polygon can be classified into *convex*, *degenerate convex* and *concave* types. A *convex* corner is a corner produced by the intersection of two consecutive sides of the polygon which form a $90°$ angle inside the interior of the polygon. A *degenerate convex* corner is produced by the intersection of two pairs of edges forming two $90°$ angles. The remaining corners are the *concave* corners, produced by the intersection of two consecutive edges of the polygon which form a $270°$ angle inside the interior of the polygon.

The interior (*resp.* boundary, corner) cover problem for a rectilinear polygon of n vertices is the following minimization problem: find a set of (possibly overlapping) rectangles of minimum cardinality so that the union of these rectangles covers the interior (*resp.* boundary, corners) of the given polygon. For the corner cover, it is sufficient that each corner is on the boundary (possibly a corner) of one of the rectangles in the given set.

An *anti-rectangle* set is a set of points inside the given rectilinear polygon such that no two of them can be covered jointly by one rectangle which does not contain a part of the exterior of the polygon. Depending upon whether it is an interior, boundary or corner cover problem, these points can be placed in the interior, boundary or corners only of the given polygon, respectively. If θ is the size of a cover for one of the cover problems, and α is the size of an anti-rectangle set for this cover, then it is obvious that $\theta \geq \alpha$. When the cover size is minimum and the size of the anti-rectangle set is maximum, $\theta = \alpha$ holds for some special cases of the cover problems. However, the equality is not true in general for either the interior, boundary or corner cover problems. Erdös asked if $\theta/\alpha \leq c$ (for some positive constant c) for the interior cover problem for arbitrary rectilinear polygons (mentioned by Chaiken et. al.[4]) and the answer is not known yet (the best known bound is $\frac{\theta}{\alpha} \leq \log \alpha$[8]).

Let A be an NP-complete minimization problem and H be a polynomial-time approximation heuristic for A. We say that H has a performance ratio of c iff for any instance I of A,

$$cost_H(I) \leq c \ . \ cost_{opt}(I)$$

where $cost_H(I)$ (resp. $cost_{opt}(I)$) denotes the cost of the solution of the instance I as obtained by H (resp., by the optimal algorithm).

For any minimization problem A let c_{opt} be the cost of the optimal solution and c_{approx} be the cost of an approximate solution produced by a heuristic. Let $\epsilon_n = \frac{|c_{opt} - c_{approx}|}{c_{opt}}$ be the relative error of the approximate solution for input size n. A polynomial-time approximation scheme for A is an algorithm that takes as input an instance of the problem and a constant $\epsilon > 0$, and produces a solution with $\epsilon_n \leq \epsilon$ in time polynomial in n[6]. A more detailed discussion of the related concepts is available in [6, 13].

2.2. Complexities of Exact Solution

In this section we summarize the previous results about the complexities of exact solutions for the interior, boundary and corner cover problems.

2.2.1. *Interior Cover Problem*

Masek[19] was the first to show that the interior cover problem is NP-complete for rectilinear polygons with holes. For a long time the complexity of this problem was unknown for polygons without holes, until Culberson and Reckhow[7] showed the interior cover problem is NP-complete even if the polygon has no holes, and even if

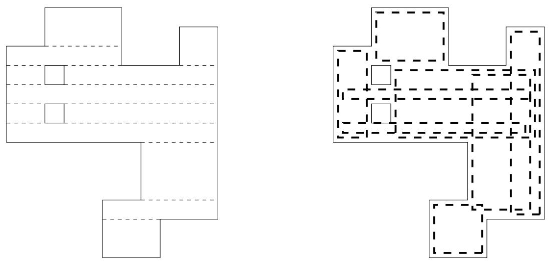


Fig. 1. *The sweepline heuristic for the interior cover problem. (a) The partition phase (b) The Extend/Delete phase. The rectangles are shown slightly offset for clarity.*

the polygon is required to be in general position. The NP-completeness reduction of Culberson and Reckhow is quite involved. They reduce the satisfiability problem to this problem. Given an instance of the satisfiability problem, they construct an instance of the interior cover problem for polygons without holes such that the interior of the constructed polygon can be covered with a specified number of rectangles if and only if the given formula is satisfiable.

2.2.2. *Boundary Cover Problem*

Conn and O'Rourke[5] showed that the boundary cover problem is NP-complete for polygons with holes, even if the polygon is in general position. As before, the complexity of the problem for polygons without holes was not known for quite some time until Culberson and Reckhow[7] showed the boundary cover problem is NP-complete even if the polygon has no holes, and even if the polygon is required to be in general position.

2.2.3. *Corner Cover Problem*

Conn and O'Rourke[5] showed that the following version of the corner cover problem is NP-complete: cover each concave corner by two rectangles along both the perimeter segments defining this corner. Using similar techniques, Berman and DasGupta[2] showed that the corner problem is NP-complete for polygons with holes. The complexity of the corner cover problem for polygons without holes is still unknown, although it is conjectured to be NP-complete by Conn and O'Rourke[5].

2.3. Approximation Heuristics

Because of the hardness of the rectilinear cover problems as stated in the previous section, it is of importance to consider efficient heuristics for the problem. Below we summarize some of the known heuristics for these problems.

2.3.1. *Interior Cover*

Franzblau[8] proposed the following sweep-line heuristic for the interior cover problem.

ALGORITHM Partition/Extend.

Input: Rectilinear polygon P.

Output: Rectangle cover for the interior of P.

- Partition P into a set of disjoint rectangles by extending each horizontal edge of P (see fig. 1(a)).
- Extend each rectangle vertically inside P until it is vertically maximal. Delete any repeated rectangles (see fig. 1(b)).

It is possible to implement the above heuristic so that it runs in $O(n \log n)$ time using standard data structures.

Let $c_{PE}(I)$ be the number of rectangles used by the above heuristic and $c_{opt}(I)$ be the optimal number of rectangles needed for the cover for an instance I. The following theorem was proved by Franzblau[8].

Theorem 2.3.1 *[8]* $c_{PE}(I) = O(c_{opt}(I). \log(c_{opt}(I)))$.

However, if the given polygon has no holes, then the heuristic performs considerably better as shown by the following theorem.

Theorem 2.3.2 *[8] If the given polygon has no holes then*

$$c_{PE}(I) \leq 2.c_{opt}(I) - 1$$

2.3.2. *Boundary Cover*

Berman and DasGupta[2] suggested a very simple heuristic for the boundary cover problem. The following theorem is proved in [2].

Theorem 2.3.3 *[2] It is possible to design a heuristic for the boundary cover problem which which runs in* $O(n \log n)$ *time and has a performance ratio of* 4.

2.3.3. *Corner Cover*

Berman and Dasgupta[2] suggested the following heuristic for the corner cover problem for polygons with holes.

ALGORITHM Corner Cover.

Input: A rectilinear polygon P, possibly with holes.

Output: A set of rectangles which together cover the corners of P.

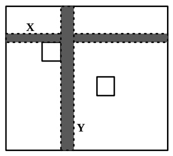


Fig. 2. *X and Y are two vertices of the graph and (X,Y) is an edge*

- Form the two sets S and T; S contains, for each horizontal segment e of P, an adjacent rectangle of width 1 of maximal horizontal extent (this is the *principal side* for this rectangle), and T contains, for each vertical segment e of P, an adjacent rectangle of width 1 of maximal vertical extent (the principal side is defined similarly). Now, form a bipartite graph $G = (S \cup T, E)$, where $E = \{(y,z) \in S \times T \mid$ principal sides of y and z share a corner $\}$ (see *fig.* 2 for an example). Construct a minimum vertex cover R of G using maximum matching. The set of rectangles R constitutes our approximate cover.

The following theorem was proved in [2].

Theorem 2.3.4 *[2] The performance ratio of the above heuristic is* 4. *It runs in* $O(n \log n)$ *time.*

When the given polygon has no holes, Berman and DasGupta[2] designed a new heuristic for the corner cover problem which runs in time $O(n \log n)$ and has a performance ratio of 2. Details of this heuristic are quite involved and available in [2].

2.4. Polynomial Time Solutions for Special Cases

In this section we survey some of the results which provide polynomial time solution for these problems when the given polygon is restricted.

2.4.1. *Interior Cover*

A rectilinear polygon P is called x-convex (resp. y-convex) if the intersection of any horizontal (resp. vertical) line segment with the interior of P is a connected (possibly empty) segment. A rectilinear polygon is *rectilinearly convex* iff it is both x-convex and y-convex. Chaiken et. al.[4] showed that for the interior cover problem for a rectilinearly convex polygon $\alpha = \theta$ (where α and θ are the sizes of maximum cardinality anti-rectangle set and minimum cardinality interior cover, respectively), and use this to provide a polynomial time solution to the interior cover problem for this special case. The result was further extended by Franzblau and Kleitman[9] who proved a similar result when the polygon is just y-convex. Lubiw[17, 18] gives polynomial time algorithms for the interior cover problem for a slightly more general class of polygons.

2.4.2. *Boundary Cover*
Using matching techniques in graphs, Conn and O'Rourke[5] gives an $O(n^{\frac{5}{2}})$ time algorithm for covering the horizontal boundary segments of a polygon P provided P is in general position. The same result can be used to devise an $O(n^{\frac{5}{2}})$ time heuristic for the boundary cover of a polygon P which has a performance ratio of 2 *provided* P is in general position.

2.4.3. *Corner Cover*
Conn and O'Rourke[5] presented an $O(n^{\frac{5}{2}})$ time algorithm for covering the convex vertices of a given polygon P optimally. However, this algorithm does not generalize to covering the concave corners of P.

2.5. Impossibility of Approximation Schemes

In this section we show that the vertex cover problem for graphs in which the degree of any vertex is bounded by a *constant* B can be reduced to the interior or boundary cover problems preserving the nature of approximation. Due to a recent result of Arora et al[1], this shows that a polynomial time approximation scheme for these covering problems is impossible, unless $P = NP$.

Let (F, P) and (G, Q) be two combinatorial optimization problems where F and G are the cost functions and P and Q are the feasibility predicates. Let $opt_{F,P}(x)$ and $opt_{G,Q}(x)$ be the optimal cost values for an instance x and $quality(a, b) = \max\left(\frac{a-b}{b}, \frac{b-a}{a}\right)$ for positive integers a and b. We say (F, P) can be reduced to (G, Q) preserving approximation[3] with amplification c if and only if there exists two deterministic polynomial time algorithms T_1 and T_2 and a positive constant c such that for all x and y, if $\bar{x} = T_1(x)$ then

(1) $Q(\bar{x}, y) \Rightarrow P(x, T_2(\bar{x}, y))$, and

(2) if $Q(\bar{x}, y)$ then $quality(opt_{F,P}(x), F(T_2(\bar{x}, y))) \leq c(quality(opt_{G,Q}(\bar{x}), G(y)))$.

This reduction ensures that an approximation G will result in an equally good approximation of F.

The *bounded-degree* vertex cover problem for graphs is as follows:

INSTANCE: An undirected graph $G = (V, E)$ with every vertex of degree at most some constant B, and a positive integer K.

QUESTION: Is there a subset $V' \subseteq V$, $\mid V' \mid \leq K$, such that for every edge $\{u, v\} \in E$, either $u \in V'$ or $v \in V'$?

Culberson and Reckhow[7] showed the NP-hardness of the rectilinear cover problem without holes by reducing the 3-SAT problem to this problem. However, their reduction is not approximation preserving since it introduces quadratically many rectangles in the optimal solution and hence does not preserve the approximation quality in the sense described above. Here, we show how to reduce the bounded-degree vertex cover problem to the interior or boundary cover problems *preserving* the approximation nature. We first consider the interior cover problem.

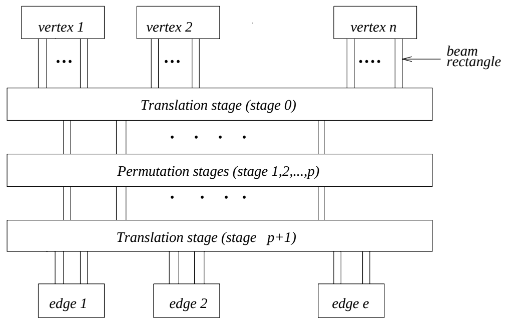

vertex 1
vertex 2
vertex n
beam rectangle
Translation stage (stage 0)
Permutation stages (stage 1,2,...,p)
Translation stage (stage p+1)
edge 1
edge 2
edge e

Fig. 3. *The overall scheme*

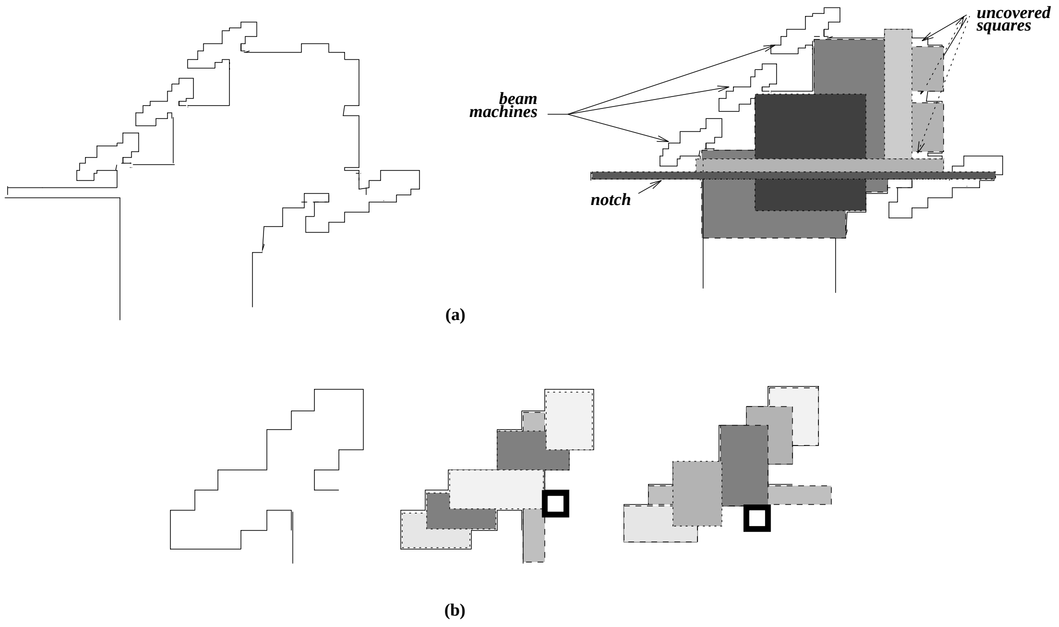


Fig. 4. *(a) A vertex gadget and its background covers and uncovered squares. (b) A beam machine and its two optimal covers. The uncovered square near its mouth is shown by thick lines.*

The overall scheme of our approach is shown in *fig.* 3. We use a *gadget* for every vertex. Beams (rectangles) coming out of a gadget indicate that this vertex participates in vertex cover. The beams are first translated, then permuted appropriately, again translated and finally enter the edge-gadgets. Each edge gadget is coupled with two beams and represents an edge between the two vertices which correspond to the two beams.

Now, we describe each component in details.

Vertex gadget: The vertex gadget is shown in full details in *fig.* 4(a). It consists of B beam machines when B is the degree of this vertex ($B = 3$ in the figure). Each beam machine can be covered optimally with 6 rectangles with only one

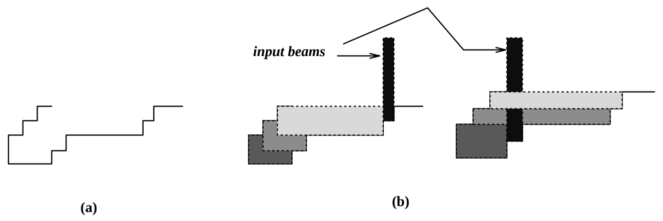


Fig. 5. *(a) An edge gadget. (b) Its two optimal covers.*

rectangle extending through its mouth in horizontal or vertical direction (see *fig.* 4(b)), and in each of these two optimal covers there is an uncovered square near the mouth of the machine (shown in *fig.* 4(b)). There is one additional beam machine at bottom right, and a notch at the extreme left bottom, which forces this beam machine to use its horizontal beam. The background of this gadget can be covered optimally with $2B+1$ rectangles, thus leaving out B uncovered squares (*fig.* 4(a)). These squares can be covered by the horizontal beams of B beam machines in an optimal cover, or by one more additional rectangle in a non-optimal cover. This structure has the following properties.

(a) There is an optimal cover of this gadget with $8B+1$ rectangles when no beam from any of the beam machines is "used" (*i. e.,* extends vertically downwards). This corresponds to the case when this vertex does not participate in a vertex cover.

(b) If a beam from any beam machine extends vertically downwards then $8B+2$ rectangles are necessary and sufficient to cover this gadget. The same property holds when more than one beam from one or more beam machines extends vertically downwards. This corresponds to the case when this vertex participates in a vertex cover.

For each of the covers described above, it is also possible to place an equal number of anti-rectangle points in the interior of the polygon.

Edge gadget: This gadget is shown in *fig.* 5. If either of its two input beams are used then it can be covered optimally with 4 rectangles, otherwise it requires at least 5 rectangles. This is same as the *inverter* structure in[7].

Translation stage: It consists of e pairs of *joints*[7] where e is the number of edges in the graph. A pair of joint rectangles is an aligned pair of beam machines (*fig.* 6). If the "incoming" beam for the left polygon of the joint is present (*i. e.* , the corresponding edge is present) then the "outgoing" beam from the right polygon of the joint should be used for optimal cover, otherwise the common horizontal rectangle between should be used. The unique background cover for this stage (covering the staircases with the uncovered squares at the mouth of the joint polygons (refer to *fig.* 4(b) for such uncovered squares)) involves $2e$ rectangles and is shown in *fig.* 6(b). This is a slightly simplified version of the joint structure in Culberson and Reckhow[7]. There are two translation stages.

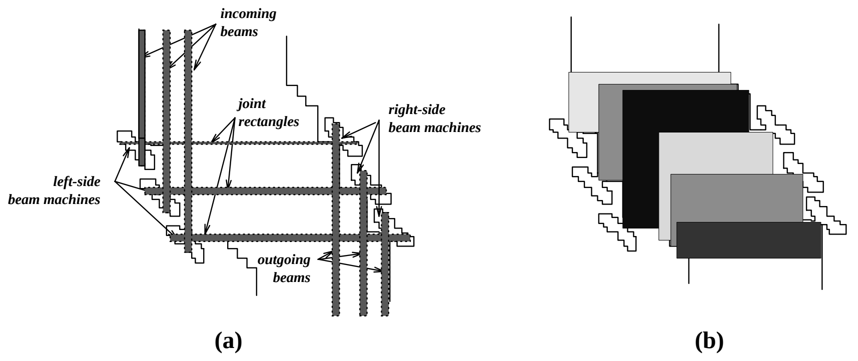


Fig. 6. *(a) A translation stage for* 3 *beams. (b) Its background covers.*

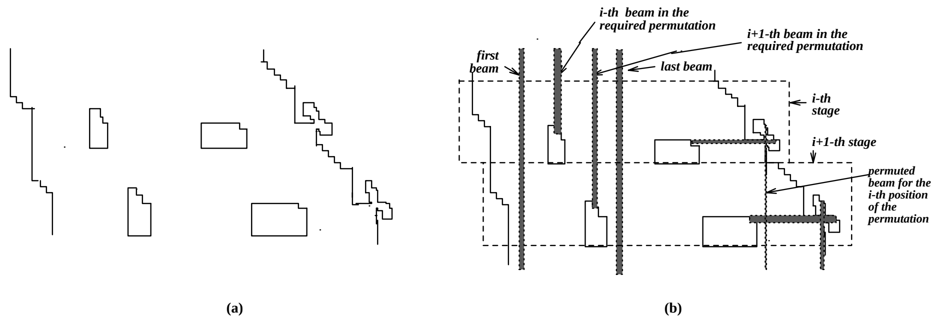

i-th beam in the required permutation
i+1-th beam in the required permutation
first beam
last beam
i-th stage
i+1-th stage
permuted beam for the i-th position of the permutation
(a)
(b)

Fig. 7. *(a) A permutation stage. (b) The beam which is permuted is shown.*

They are used to allow the optimal covering of permutation stages not to affect the vertex gadgets and the edge gadgets.

Permutation stage: There are at most $p \leq 2e$ permutation stages when e is the number of edges in the graph. These are needed because the order in which the beams come out of the vertex gadgets is not necessarily the same as they should arrive at the edge gadgets. Each stage consists of staircases and holes as shown in *fig.* 7(a). In stage i we put the i^{th} beam from left "in the required permutation" in its correct place and so the boundary of the polygon is always visible from the right-side hole. The right-side hole is placed so that it makes impossible the right-staircases of the previous stages to be covered by any rectangle which covers optimally the background this or later stages. If a beam is present and covers one notch of the left-side hole, the vertical beam of the beam machine at the right is used for the optimal cover, otherwise, for optimal cover, the horizontal beam of this beam machine covers the notch of the right-side hole (and, so, its vertical beam cannot be used for optimal cover). In all we need 8 rectangles to cover each stage, because, there are 8 anti-rectangle points, and none of these can be covered by rectangles covering those of previous or later stages. The left-side hole can be merged with the boundary for stage n. The details are given in a lemma below.

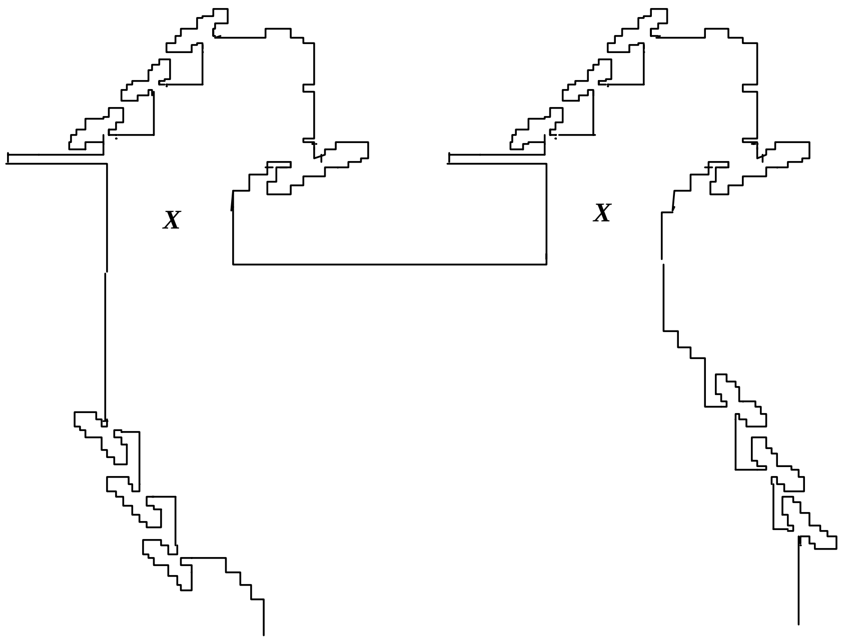


Fig. 8. *Joining vertex gadgets (for a graph having 2 vertices) to the translation stage (anti-rectangle points marked by* **X***).*

The proofs of the following lemma follow from the above discussion.

Lemma 2.5.1 *A vertex gadget can be covered optimally only if all of its beam rectangles are not "used".*

Lemma 2.5.2 *The part of the polygon connecting the vertex gadgets to the translation stage needs v rectangles to cover it independent of the cover of any other stage or gadgets, where v is the number of vertices of the graph and these rectangles do not influence the optimal cover of the total polygon.*

Proof. We can place anti-rectangle points at each joints of vertex gadgets to the translation stage, and none of these anti-rectangle points can be covered together with those in the covers of the vertex gadgets, the translation stage or any other stages (see *fig.* 8).□

The proofs of the following two lemmas are straightforward from the discussion above.

Lemma 2.5.3 *For each translation stage the following are true:*

(a) *It requires $2e$ rectangles to cover its staircases along with the mouth of the joint polygons (i. e., its background).*

(b) *If the incoming beam to the left-side polygon is present, then for optimal cover the outgoing beam of the right-side polygon should be present. However, if the*

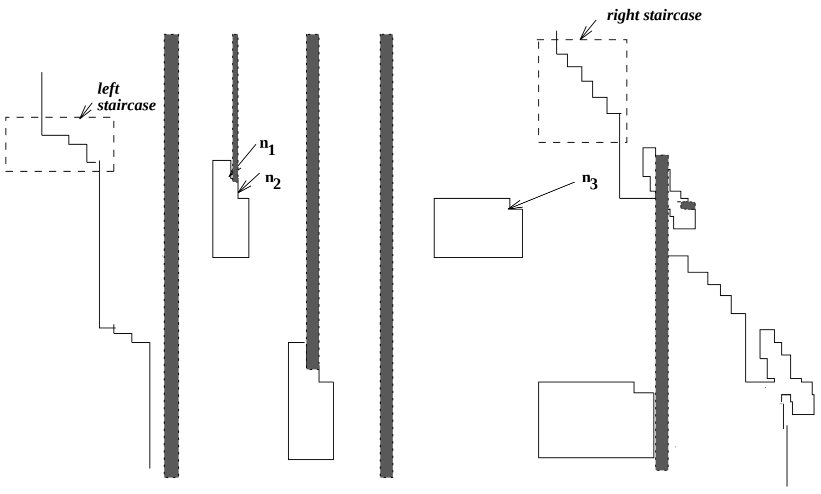


Fig. 9. *Permutation stages. The incoming beam is shown to be present.*

input beam is not present, then for optimal cover the joint rectangle must be present.

(c) *The joint polygons can always be covered optimally, if the rule of (b) above is followed.*

(d) *The optimal cover of the background of this stage cannot be affected by a rectangle that participates in an optimal cover of vertex or edge gadgets or other stages.*

Lemma 2.5.4 *Consider the i^{th} permutation stage (stage i). Then the following are true (fig. 9):*

(a) *Stage i has 8 anti-rectangle points (for its background cover) which cannot be covered together with those of any other stage. Also, 8 rectangles are sufficient to cover background of stage i.*

(b) *Let the two notches of the left hole be $n1$ and n_2 and that of right hole be n_3. Then,*

(i) *if n_1 is covered by the incoming beam, then for optimal cover n_2, n_3 and the three left staircases must be covered by rectangles which cover the 5 right staircases and hence the right-side beam machine can send out its vertical beam.*

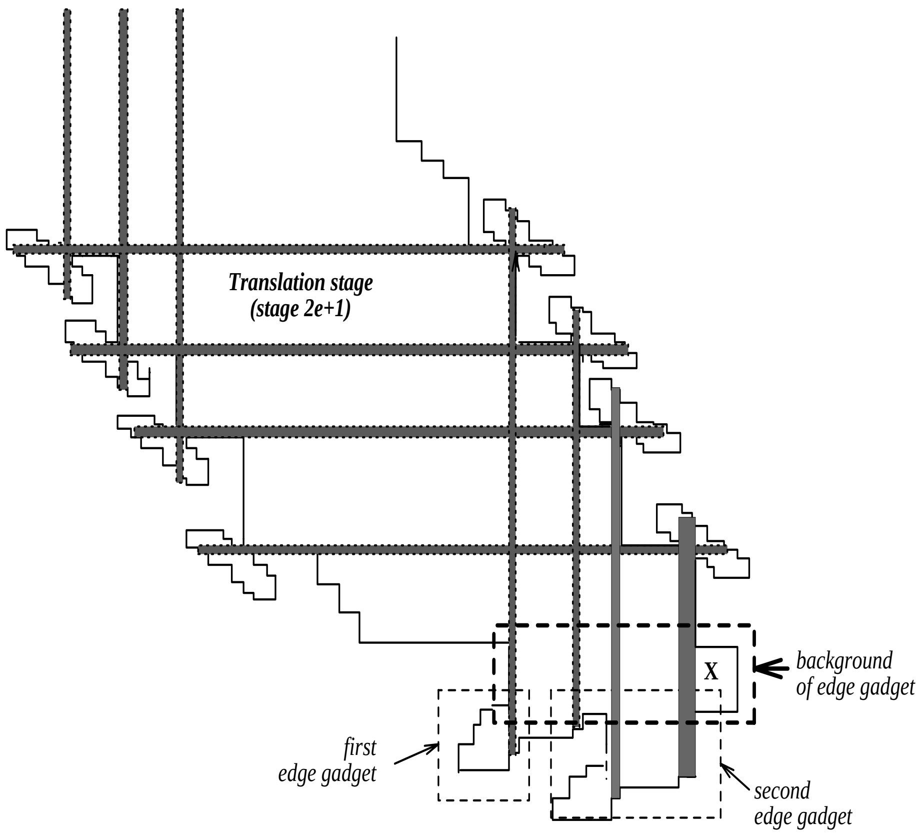


Fig. 10. *Edge gadgets for a graph having 2 edges. The anti-rectangle point in the background cover of the edge is shown by* **X**.

> **(ii)** *if n_1 is not covered by the incoming beam, then for optimal cover notches n_1, n_2 and the three left staircases must be covered by the rectangles which cover the 5 right staircases and n_3 is covered by the horizontal beam of the beam machine (and hence the outgoing beam is absent).*

(c) *Anti-rectangle points for covering the background of stage i cannot be covered together with those of any other stage.*

(d) *The stage i places the i^{th} beam from the left in the required permutation into its correct position.*

Lemma 2.5.5 *The background of the edge gadget needs 1 rectangle to cover optimally independent of any other stage. This optimal cover of the background of the edge gadgets is not influenced by the covering rectangles of the previous stages. If* *an edge gadget gets one or two incoming beams, it needs only* 4 *more rectangles to cover it.*

Proof. Rectangles covering vertex gadgets or stages 1 to $2e$ cannot extend to the edge gadget. The beam rectangles in stage $2e + 1$ can extend to cover edge gadgets (see *fig.* 10), but these rectangles cannot take part in the optimal cover of the background of stage $2e + 1$. One rectangle is sufficient to cover the background of the edge gadget, and the corresponding anti-rectangle point is shown in *fig.* 10. □

Lemma 2.5.6 *Any covering of the polygon (corresponding to a given graph) can be transformed in polynomial time (in the size of the given graph) to another cover with the following properties without increasing the number of covering rectangles:*

(a) *Either all the beams of a vertex gadget are used or none is used.*

(b) *Each edge gadget uses at least one of its two beams from the previous translation stage (and hence needs* 4 *additional rectangles for its optimal cover).*

(c) *The background of the vertex gadgets, translation stages, permutation stages and edge gadgets are covered with the optimal number of rectangles.*

Proof.

(a) If only some beams are used for a vertex gadget we can as well use all the beam rectangles without increasing the number of rectangles for the vertex gadget. Then, we keep moving from one stage to another till we arrive at the edge gadget, turning the joint rectangle of each stage off and using the incoming and outgoing beam rectangles (if this was not the case, the cover for a particular stage was not optimal and we actually improve, refer to lemma 2.5.3, 2.5.4). Surely we do not use more rectangles at the edge gadget (we will improve if it had no incoming beam rectangles).

(b) Assume this is not the case. We keep moving from the edge gadget through successive stages towards the vertex gadgets in a manner similar to as described in (a) above (we arbitrarily turn on one of the two beams of the edge gadget). We may use one more rectangle at the vertex gadget, if it was not using its beams, but the gain of one rectangle at the corresponding edge gadget compensates.

(c) Once we have done the transformations needed for parts (a) and (b) above, we can select the necessary background covers depending on whether the incoming beams of a particular stage are used or not as outlined in lemma 2.5.2, 2.5.3 (part (a)), 2.5.4 and 2.5.5, since the optimal covers of these parts of the polygon are independent of each other and depends only on the presence or absence of the beam rectangles.

It is obvious that traversal in (a), (b) or (c) above takes polynomial time.□

Using lemmas 2.5.2, 2.5.3, 2.5.4, 2.5.5 and 2.5.6, and the properties of the gadgets, it is possible to prove the following lemma (see [2] for a proof).

Lemma 2.5.7 *Let $G = (V, E)$ be a graph in which the degree of any vertex is at most a constant $B > 0$ and P be the polygon constructed from it by the above procedure. Let the number of permutation stages needed be p ($p \leq B \cdot \mid V \mid$). Then, the following holds:*

(a) *A minimal vertex cover of G of size m corresponds to a rectilinear cover of P of size at most $30 \cdot \mid E \mid +14 \cdot p + (8 \cdot B + 2) \cdot \mid V \mid +m + 1$.*

(b) *A rectilinear cover of P, after the transformation as outlined in lemma 2.5.6, of size θ corresponds to a minimal vertex cover of size at least $\theta - (30 \cdot \mid E \mid +14 \cdot p + (8 \cdot B + 2) \cdot \mid V \mid +1)$.*

Theorem 2.5.1 *The reduction of the vertex cover problem to the rectilinear cover problem as outlined above is approximation preserving.*

Proof. T_1 is the transformation needed to construct the polygon using gadgets as outlined above. The transformation T_2 consists of the following:

(a) Modify the cover as outlined in lemma 2.5.6,

(b) Select a vertex in the vertex cover if and only if the corresponding gadget in the polygon uses all its beam rectangles.

The quality constraint for the reduction follows from lemma 2.5.7 above, and the fact that $p \leq 2 \cdot \mid E \mid \leq B \cdot \mid V \mid$. □

Hence, we have proved the following result.

Theorem 2.5.2 *No polynomial-time approximation scheme exists for the interior cover problem, unless P=NP.*

A careful examination of our construction shows that all the results hold for boundary cover also, in particular we can always place all the anti-rectangle points on the boundary of the polygon. Hence, we also prove the following result.

Theorem 2.5.3 *No polynomial-time approximation scheme exists for the boundary cover problem, unless P=NP.*

3. Segmented Channel Routing

3.1. Motivation

The architecture of channeled FPGAs [12] is similar to that of conventional (mask programmed) gate arrays: comprising rows of logic cells separated by segmented routing channels (Fig. 11). The inputs and outputs of the cells each connect to a dedicated vertical segment. Programmable switches are located at each crossing of vertical and horizontal segments and also between pairs of adjacent horizontal segments in the same track. By programming a switch, a low resistance path is created between the two crossing or adjoining segments.

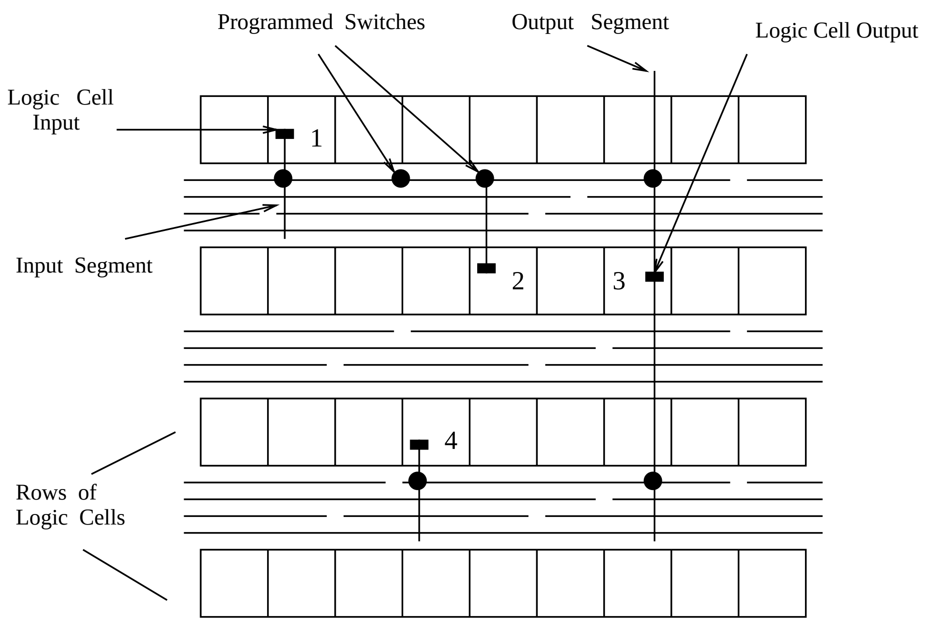


Fig. 11. FPGA routing architecture. • denotes a programmed switch; unprogrammed switches are omitted for clarity.

A typical example of routing in a channeled FPGA is shown in Fig. 11. The vertical segment connected to the output of cell 3 is connected by a programmed switch to a horizontal segment, which in turn is connected to the input of cell 4 through another programmed switch. In order to reach the inputs of cells 1 and 2, two adjacent horizontal segments are connected to form a longer one.

The choice of the wiring segment lengths in a segmented channel is driven by tradeoffs involving the number of tracks, the resistance of the switches, and the capacitances of the segments. These tradeoffs are illustrated in Fig. 12.

Fig. 12(a) shows a set of connections to be routed. With the complete freedom to configure the wiring afforded by mask programming, the *left edge* algorithm [14] will always find a routing using a number of tracks equal to the density of the connections (Fig. 12(b)). This is the case since there are no 'vertical constraints' in the problems we consider.

In an FPGA, achieving this complete freedom would require switches at every cross point. Furthermore, switches would be needed between each two cross points along a wiring track so that the track could be subdivided into segments of arbitrary length (Fig. 12(c)). Since all present technologies offer switches with significant resistance and capacitance, this would cause unacceptable delays through the routing. Another alternative would be to provide a number of continuous tracks large enough to accommodate all nets (Fig. 12(d)). Though the resistance is limited, the capacitance problem is only compounded, and the area is excessive.

A segmented routing channel offers an intermediate approach. The tracks are divided into segments of varying lengths (Fig. 12(e)), allowing each connection to be routed using a single segment of the appropriate size. Greater routing flexibility is obtained by allowing limited numbers of adjacent segments in the same track to be joined end-to-end by switches (Fig. 12(f)). Enforcement of simple limits on the number of segments joined or their total length guarantees that the delay will not be unduly increased. Our results apply to the models of Fig. 12(e) and Fig. 12(f).

In Section 3.2 we formally define segmented channel routing and summarize the key results. Details of some of the algorithms are given in Section 3.3.

3.2. Definitions and Survey of Results

The input to a segmented channel routing problem, as depicted in Fig. 13, is a segmented channel consisting of a set $\mathcal{T}$,of T tracks, and a set $\mathcal{C}$ of M connections. The tracks are numbered from 1 to T. Each track extends from column 1 to column N, and is divided into a set of contiguous segments separated by switches. The switches are placed between two consecutive columns.

For each segment s, we define $left(s)$ and $right(s)$ to be the leftmost and rightmost column in which the segment is present, $1 \leq left(s) \leq right(s) \leq N$. Each connection c_i, $1 \leq i \leq M$, is characterized by its left-most and right-most column: $left(c_i)$ and $right(c_i)$. Without loss of generality, we assume throughout that the connections have been sorted so that $left(c_i) \leq left(c_j)$ for $i < j$.

A connection c may be *assigned* to a track t, in which case the segments in track t that are present in the columns spanned by the connection are considered *occupied.* More precisely, a segment s in track t is occupied by the connection c if $right(s) \geq left(c)$ and $left(s) \leq right(c)$. In Fig. 13 for example, connection c_3 would occupy segments s_{21} and s_{22} in track 2 or segment s_{31} in track 3.

Definition 3.2.1 [**Routing**]
A routing, R, of a set of connections is an assignment of each connection to a track such that no segment is occupied by more than one connection.
A **K-segment** routing is a routing that satisfies the additional requirement that each connection occupies at most K segments.

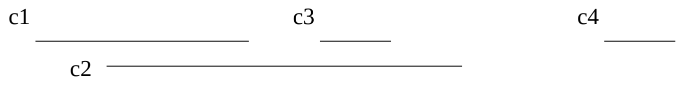
We can now define the following segmented channel routing problems:

Problem 3.2.1 [**Unlimited Segment Routing**] Given a set of connections and a segmented channel, find a routing.

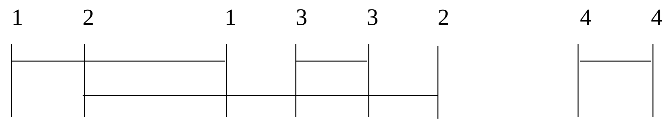
To reduce the delay through assigned connections, it may be desirable to limit the number of segments used for each connection.

Problem 3.2.2 [**K-Segment Routing**] Given a set of connections and a segmented channel, find a K-segment routing.

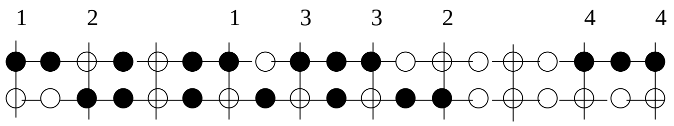
It is often desirable to determine a routing that is optimal with respect to some criterion. We may thus specify a weight $w(c,\ t)$ for the assignment of connection c to track t, and define:



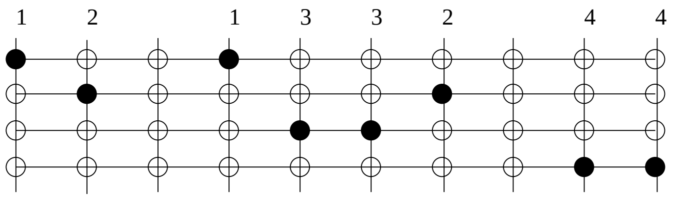
a. Set of connections to be routed.



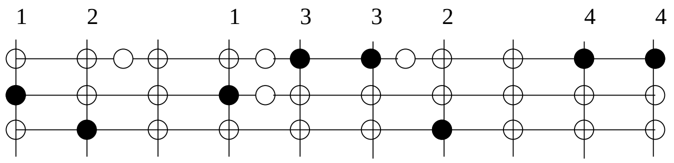
b. Routing in unconstrained channels.



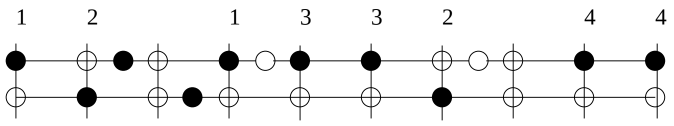
c. Routing in fully segmented channels.



d. Routing in unsegmented channel.



e. Segmented for 1-segment routing.



f. Segmented for 2-segment routing.

Fig. 12. Examples of channel routing. ∘ denotes open switch; • closed switch.

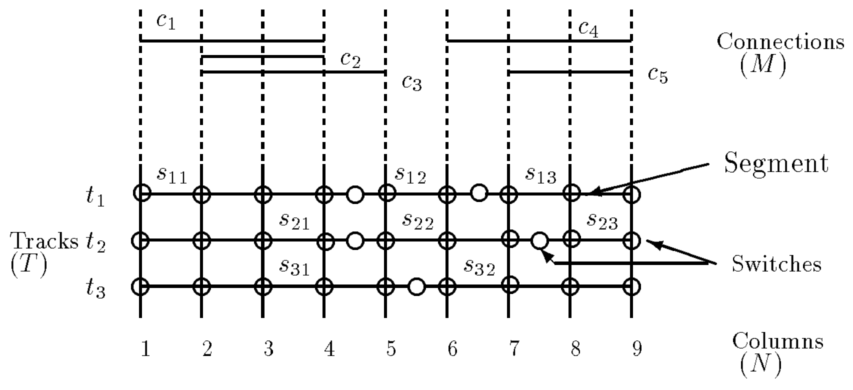


Fig. 13. An example of a segmented channel and a set of connections. $M = 5$, $T = 3$, $N = 9$. Connections: c_1, c_2, c_3, c_4, c_5. Segments: s_{11}, s_{12}, s_{13}, s_{21}, s_{22}, s_{23}, s_{31}, s_{32}.

Problem 3.2.3 [**Optimal Routing**] Given a set of connections and a segmented channel, find a routing which assigns each connection c_i to a track t_i such that $\sum_{i=1}^{M} w(c_i, t_i)$ is minimized.

For example, a reasonable choice for $w(c,t)$ would be the sum of the lengths of the segments occupied when connection c is assigned to track t. Note also that with appropriate choice of $w(c,t)$, Problem 3.2.3 subsumes Problem 3.2.2.

The problems defined above consider segmented channel routing with the restriction that each connection may only be assigned to a single track. It is easy to see that the routing capacity of a segmented channel may be increased if a connection is assigned to segments in different tracks. For example, consider the segmented channel routing problem in Fig. 14. It can be easily shown that if the assignment of each connection is constrained to a single track successful routing does not exist. However, by assigning connection c_2 to segments s_{11} and s_{33}, which are located in tracks t_1 and t_3, successful routing may be achieved. We refer to such a routing as generalized routing.

Definition 3.2.2 [**Generalized Routing**]
A generalized routing, R_G, of a set of connections consists of an assignment of each connection to one or more tracks such that no segment is occupied by more than one connection.

Thus a generalized routing allows each connection $c = (left(c), right(c))$ to be *split* into p $(p \geq 1)$ parts: $(left(c), l_1)$, (l_1+1, l_2), $(l_2+1, l_3), \cdots$, $(l_{p-1}+1, right(c))$, such that each part can be assigned to different tracks. A column l_i, where a connection is split, is referred to as a column where the connection c *changes* tracks.

Detailed hardware implementations may be developed to support generalized routing. For example, vertical wire segments may be added to facilitate track changing. In this case if a connection changes tracks, two switches must be programmed

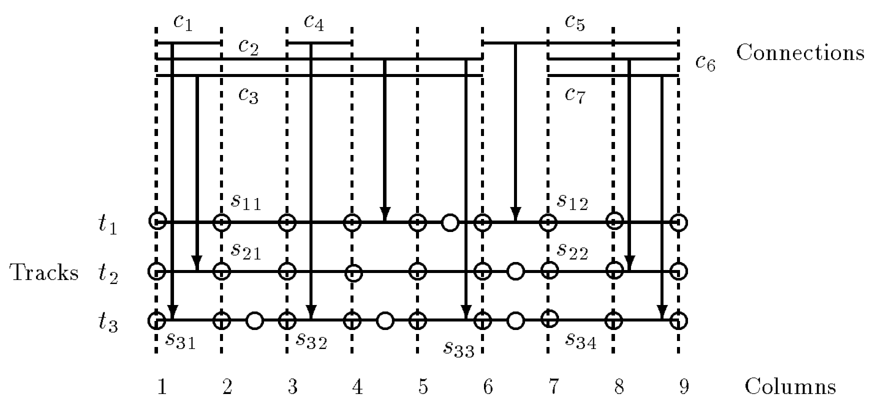


Fig. 14. An example where generalized routing is necessary for successful assignment.

compared to only one if the connection is assigned to two contiguous segments in the same track. Thus allowing connections to occupy multiple tracks might lead to increase in area and to greater delays.

Motivated by such penalties, constraints may be imposed on the generalized segmented channel routing problem leading to the following potentially important special cases:

1. Determine a generalized routing that uses at most k segments for routing any particular connection.
2. Determine a generalized routing that uses at most l different tracks for routing any connection.
3. Determine a generalized routing where connections can switch tracks only at predetermined columns.

Preliminary results on the following unconstrained version of generalized segmented channel routing problem is presented in [25].

Problem 3.2.4 [**Generalized Segmented Channel Routing**] Given a set of connections and a segmented channel, find a generalized routing.

In [25] the following results are presented.

Theorem 3.2.1 Determining a solution to Problem 3.2.1 is strongly NP-complete.

Theorem 3.2.2 Determining a solution to Problem 3.2.2 is strongly NP-complete even when $K = 2$.

The reductions used to prove these theorems are rather tricky, and may have applications to problems in the area of task-scheduling on non-uniform processors.

Although Theorems 3.2.1 and 3.2.2 show that segmented channel routing is in general NP-complete, several special cases of the problem are tractable. In this chapter we present polynomial-time algorithms for the following special cases:

Identically Segmented Tracks: Two tracks will be defined to be identically segmented if they have switches at the same locations, and hence, segments of the same length. The *left edge* algorithm used for conventional channel routing can be applied to solve Problems 3.2.1, 3.2.2, and 3.2.3.
1-Segment Routing: A routing can be determined by a linear time ($O(MT)$) greedy algorithm that exploits the geometry of the problem. The corresponding optimization problem can be also solved in polynomial time by reducing it to a weighted maximum bipartite matching problem.
At Most 2-Segments Per Track: If each track is segmented into at most two segments then also a greedy linear time algorithm (similar to the one for 1-Segment routing) can be designed to determine a routing.

We have also developed a general $O(T!M)$-time algorithm using dynamic programming for solving Problems 3.2.1, 3.2.2, and 3.2.3. This general algorithm can be adapted to yield more efficient algorithms for the following cases:
Fixed Number of Tracks: If the number of tracks is fixed then the general algorithm directly yields a polynomial time algorithm.
K-segment Routing: The general algorithm can be modified to yield an $O((K+1)^T M)$-time algorithm. Note that for small values of K the modified algorithm performs better than the general one.
Fixed types of Tracks: If the number of tracks is *unbounded* but the tracks are chosen from a fixed set, where T_i is the number of tracks of type i, then an $O((\prod_1^l T_i^{K+2})M)$ time (hence, a polynomial-time) algorithm can be designed.

3.3. Algorithms

In this section we present algorithms for various special cases of Problems 3.2.1-3.2.3. We first discuss algorithms that exploit the geometry of the segmented channels. We then discuss a general algorithm based on dynamic programming. Finally we discuss a heuristic algorithm (based on linear programming) that appears to work surprisingly well in practice.

3.3.1. *Geometrical Algorithms*

Identically Segmented Tracks

If all tracks are identically segmented (i.e., the locations of the switches are the same in every track), then Problems 3.2.1 and 3.2.2 can be solved by the left edge algorithm [14] in time $O(MT)$. Assign the connections in order of increasing left ends as follows: assign each connection to the first track in which none of the segments it would occupy are yet occupied.

Note that the density of the connections does not provide an upper bound on the number of tracks required for routing (as is the case for conventional routing when the left edge algorithm is used in the absence of vertical constraints). However, if prior to computing the density, the ends of each connection are extended until a column adjacent to a switch is reached, then the density would be a valid upper bound.

1-Segment Routing

If we restrict consideration to 1-segment routings, Problem 3.2.2 can be solved by the following greedy algorithm.

> The connections are assigned in order of increasing left ends as follows. For each connection, find the set of tracks in which the connection would occupy one segment. Eliminate any tracks where this segment is already occupied. From among the remaining tracks, choose one where the unoccupied segment's right end is closest to the left (i.e. the right end coordinate of the segment in the chosen track is the smallest) and assign the connection to it. If there is a tie, then it is broken arbitrarily. In the example of Fig. 13, the algorithm assigns c_1 to s_{11}, c_2 to s_{21}, c_3 to s_{31}, c_4 to s_{32}, and c_5 to s_{13}. The time required is $O(MT)$.

Next we show that if some connection cannot be assigned to any track, then no complete routing is possible. The proof of the following theorem can be found in [25].

Theorem 3.3.1 The above algorithm solves Problem 3.2.2 if $K = 1$.

For 1-segment routing, Problem 3.2.3 may be solved efficiently by reducing it to a bipartite matching problem. The underlying bipartite graph can be constructed as follows: the left side has a node for each connection and the right side a node for each segment. An edge is present between a connection and a segment if the connection can be assigned to the segment's track. The weight $w(c,\ t)$ is assigned to the edge between connection c and a segment in track t. A minimum-weight matching indicates an optimal routing. The time required using the best known matching algorithm (see [24]) is $O(V^3)$, where $V \leq M + NT$ is the number of nodes.

At Most 2-Segments Per Track

In a track with 2-segments, the first segment from the left will be referred to as the *initial* segment and the next one will be referred to as the *end* segment. If the track is unsegmented, i.e. it has only one segment, then for our purposes we will refer to the only segment as an end segment.

The following greedy algorithm, which is similar to the one for 1-Segment routing, can be used to determine a solution to Problem 3.2.1:

> The connections are assigned in order of increasing left ends (ties are resolved arbitrarily). During the execution of the algorithm a track will be considered as *unoccupied* if no connection has been assigned to it.
>
> Now for each connection, determine the set of tracks in which the connection would occupy a single segment. Eliminate any track where this segment is already *occupied*. Now consider the following two cases:
> Case 1. If no track is available (i.e. after the above mentioned elimination of tracks) then append the connection to the pool, P, of unassigned (but already examined) connections.
> Case 2. If tracks are available then assign the connection to a track where the unoccupied segment's right end is closest to the left (i.e. the right end coordinate of the segment in the chosen track is the smallest). If more than one track qualifies then the tie is broken arbitrarily.

Next, if $|P|$ (i.e. the number of unassigned, but already examined, connections) equals the number of tracks unoccupied by any connection, then assign the connections in P to these unoccupied tracks in any order; mark these tracks as occupied, and remove the assigned connections from P. Else, if $|P|$ is greater than the number of such unoccupied tracks then stop, and signal that no valid routing is possible.

Continue with the next connection.

When, all the connections are examined and pool P is non-empty then assign the connections in P to unoccupied tracks.

Theorem 3.3.2 The above mentioned algorithm determines a routing, if there exists one, for the case where every track has at most two segments.

A proof for the above theorem is outlined in [25].

3.3.2. *A General Algorithm for Determining Routing*

Although the problem of determining a routing for a given segmented channel and a set of connections is in general NP-complete, we describe below an algorithm that finds a routing in time linear in M (the number of connections) when T (the number of tracks) is fixed. This is of interest since T is often substantially less than M. The algorithm may also be quite efficient when there are many tracks, but they are segmented in a limited number of ways (see Theorem 7 below). The algorithm first constructs a data structure called an *assignment graph* and then reads a valid routing from it. The same algorithm applies to both Problems 3.2.1 and 3.2.2, though with different time and memory bounds. It can also be extended to Problem 3.2.3.

Frontiers and the Assignment Graph

Given a valid routing for connections c_1 through c_i, it is possible to define a *frontier* which constitutes sufficient information to determine how the routing of $c_1 ... c_i$ may be extended to include an assignment of c_{i+1} to a track such that no segment occupied by any of c_1 through c_i will also be occupied by c_{i+1}. Fig. 15 shows an example of a frontier. It will be apparent that c_{i+1} may be assigned to any track t in which the frontier has not advanced past the left end of c_{i+1}. For example, in Fig. 15 connection c_4 can be assigned to track t_2 but not to track t_1.

More precisely, given a valid routing of $c_1, \cdots, c_i$, $1 \leq i < M$, define the frontier $\mathbf{x}$ to be a T-tuple ($\mathbf{x}[1]$, $\mathbf{x}[2]$, ..., $\mathbf{x}[T]$) where $\mathbf{x}[j]$ is the leftmost unoccupied column in track t_j at or to the right of column $left(c_{i+1})$. (A column in track t_j is considered unoccupied if the segment present in the column is not occupied.) The frontier is thus a function $\mathbf{x} = F_i(t_{c_1}, \cdots, t_{c_i})$ of the tracks $t_{c_1}, \cdots, t_{c_i}$ to which $c_1 ... c_i$ are respectively assigned. For $i = 0$, let $\mathbf{x} = F_0$, where $F_0[t] = left(c_1)$ for all t. For $i = M$, let $\mathbf{x} = F_M$, where $F_M[t] = N + 1$ for all t.

Next, we describe a graph called the assignment graph which is used to keep track of partial routings and the corresponding frontiers. A node at level i, $1 \leq i < M$, of the assignment graph corresponds to a frontier resulting from some valid routing of c_1 through c_i. Level 0 of the graph contains the root node, which corresponds to F_0. If a complete valid routing for $c_1, \cdots, c_M$ exists, then level M of the graph contains a single node corresponding to F_M. Otherwise, level M is empty.

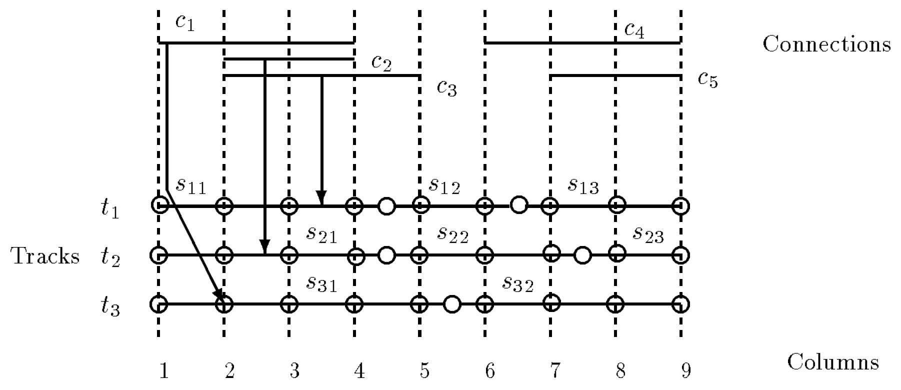


Fig. 15. A frontier for the example of Fig. 13. Connections c_1, c_2, and c_3 are assigned to segments s_{31}, s_{21}, and $\{s_{11}, s_{12}\}$ respectively. The frontier is $\mathbf{x} = [7, 6, 6]$.

The assignment graph is constructed inductively. Given level $i \geq 0$ of the graph, construct level $i + 1$ as follows. (For convenience, we identify the node by the corresponding frontier.)

```
For each node $\mathbf{x}_i$ in level $i$ {
        For each track $t_j$, $1 \leq j \leq T$ {
              If $\mathbf{x}_i[j] = left(c_{i+1})$ {
                    /* $c_{i+1}$ can be assigned to track $t_j$. */
                    Let $\mathbf{x}_{i+1}$ be the new frontier after $c_{i+1}$ is assigned to track $t$.
                    If $\mathbf{x}_{i+1}$ is not yet in level $i + 1$ {
                          Add node $\mathbf{x}_{i+1}$ to level $i + 1$.
                          Add an edge from node $\mathbf{x}_i$ to node $\mathbf{x}_{i+1}$. Label it with
$t_j$.
                          }
                    }
              Else {
                    /* $\mathbf{x}_i[j] > left(c_{i+1})$ so $c_{i+1}$ cannot be assigned to track $t_j$. */
              Continue to next track $t_{j+1}$.
              }
        }
}
```

If there are no nodes added at level $i + 1$, then there is no valid assignment of c_1 through c_{i+1}.

Searching for the node $\mathbf{x}_{i+1}$ in level $i + 1$ can be done in $O(T)$ time using a hash table. Insertion of a new node in the table likewise requires time $O(T)$.

If there are a maximum of L nodes at each level, then construction of the entire assignment graph requires time $O(MLT^2)$. Once the assignment graph has been constructed, a valid routing may be found by tracing a path from the node at level M back to the root, reading the track assignment from the edge labels. (If there is no node at level M, then no complete valid assignment exists.) This takes only $O(M)$

time, so the overall time for the algorithm is $O(MLT^2)$. The memory required to store the assignment graph is $O(MLT)$.

A minor change allows us to solve the optimization problem as well. Each edge is labeled with the weight $w(c, t_c)$ of the corresponding assignment. Each node is labeled with the weight of its parent node plus the weight of the incoming edge. The algorithm is modified as follows. If a search in level $i+1$ finds that the new node $\mathbf{x}_{i+1}$ already exists, we examine its weight relative to the weight of node $\mathbf{x}_i$ plus $w(c_{i+1},\ t_{c_{i+1}})$. If the latter is smaller, we replace the edge entering $\mathbf{x}_{i+1}$ with one from $\mathbf{x}_i$ and update the weights accordingly. Thus the path traced back from the node at level M will correspond to a minimal weight routing. The order of growth of the algorithm's time remains the same, as does that of its memory.

Analysis for Unlimited Segment Routing

The following theorem shows that for unlimited segment routing, $L \leq 2\ T!$, so that the time to construct the assignment graph and find an optimal routing is $O(M\ T^2\ T!)$ and the memory required is $O(M\ T\ T!)$. A proof of the theorem can be found in [25].

Theorem 3.3.3 For unlimited segment routing, the number of distinct frontiers that may occur for some valid assignment of c_1 through c_i is at most $2\ T!$.

Analysis for K-Segment Routing

The following theorem shows that for K-segment routing, $L \leq (K+1)^T$, so that the time to construct the assignment graph and find an optimal routing is $O(MT^2(K+1)^T)$ and the memory required is $O(MT(K+1)^T)$ (see [25] for a proof).

Theorem 3.3.4 for K-segment routing, the number of distinct frontiers that may occur for some valid routing of c_1 through c_i is at most $(K+1)^T$.

Case of Many Tracks of a Few Types

Suppose the T tracks fall into two types, with all tracks of each type segmented identically. Then two frontiers that differ only by a permutation among the tracks of each type may be considered equivalent for our purposes in that one frontier can be a precursor of a complete routing if and only if the other can. Thus we can restrict consideration to only one of each set of equivalent frontiers, and strengthen the result of Theorem 6 as follows.

Theorem 3.3.5 Suppose there are T_1 tracks segmented in one way, and $T_2 = T - T_1$ segmented another way. The number of distinct frontiers $\mathbf{x}$ that may occur for some valid K-segment routing of c_1 through c_i, and that satisfy $\mathbf{x}[i] \leq \mathbf{x}[j]$ for all $i < j$ with tracks t_i and t_j of the same type, is $O((T_1 T_2)^K)$.

A proof for the above theorem can be found in [25]. It follows that a K-segment routing may be found in time $O(M(T_1T_2)^K T^2)$, and memory $O(M(T_1T_2)^K T)$. The result of Theorem 3.3.5 may easily be generalized to the case of l types of tracks, in which case the time is $O(M(\prod_1^l T_i^K))$, and the memory is $O(M(\prod_1^l T_i^K)T)$.

3.3.3. *A Linear Programming Approach*

Problems 3.2.1 and 3.2.2 can be reduced to 0–1 Linear Programming (LP) problems via a straight-forward reduction procedure. The 0–1 LP is in general NP-Complete. For our purposes, however, such a reduction is interesting because simulations in [25] showed that for almost all cases the corresponding 0–1 LP problems could be solved by viewing them as ordinary LP problems for which efficient algorithms are known.

We now briefly describe the reduction procedure for Problem 3.2.1. The corresponding reduction for Problem 3.2.2 follows after minor modifications. Let us define binary variables x_{ij}, for $1 \leq i \leq M$, and $1 \leq j \leq T$, as follows: if $x_{ij} = 1$, then connection c_i is assigned to track t_j, else if $x_{ij} = 0$, then connection c_i is not assigned to track t_j. Since in a routing each connection is assigned to at most one track, one has the following constraints:

$$\sum_{j=1}^{T} x_{ij} \leq 1; \quad \forall \; 1 \leq i \leq M$$

One also has to make sure that in any routing two connections assigned to the same track must not share a segment. Consider a track t_j; one can then easily determine sets of connections $P_{j1}, \cdots, P_{jl_j}$ (not necessarily disjoint) such that at most one from each set can be assigned to the track t_j. Hence for each such set P_{jk}, one must satisfy

$$\sum_{c_i \in P_{jk}} x_{ij} \; \leq \; 1$$

Finally, one must make sure that all the connections are routed, this can be ensured by maximizing the following objective function:

$$\sum_{i=1}^{M} \sum_{j=1}^{T} x_{ij}$$

One can now easily verify that the above $0-1$ LP's objective function achieves the value of M if and only if there is a solution to Problem 3.2.1 [25].

4. Conclusion and Open Problems

We have discussed some of the important results for the rectilinear polygon cover problems. However, the following problems still remain open and may be worth investigating further:

- Can we prove a better upper bound for the performance ratio of the sweepline heuristic (or any other heuristic) for the interior cover problem for polygons with holes? There is currently no example known in which the sweepline heuristic has a performance ratio more than 3.
- Prove or disprove if $\frac{\theta}{\alpha} \leq c$ for some positive constant c for the interior cover problem.
- Prove or disprove that the corner cover problem is NP-complete for rectilinear polygons without holes.

We have also introduced problems concerning the design and routing for segmented channels. There are several open issues in this new area of routing. For example, although efficient algorithms for many special cases of the routing problem have been developed, several other interesting cases are yet to be solved; following are some relevant ones: 1. connection lengths are bounded; and 2. connections are non-overlapping. An important open problem is to develop efficient algorithms for approximate segmented channel routing. Also, efficient algorithms for the generalized routing problems are not known.

Acknowledgements

The first author wishes to thank Prof. Piotr Berman with whom part of the research work on this problem was carried out and Prof. Georg Schnitger for suggesting the problem of non-approximability of the interior cover problem. The second author would like to thank Prof. Abbas El Gamal and Dr. Jonathan Greene with whom parts of the research on segmented channel routing was carried out.

References

1. S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof Verification and Hardness of Approximation Problems. *Proc. 33rd IEEE Symp. Found. of Comp. Sc.(FOCS)* (1992), pp. 14-23.
2. P. Berman and B. DasGupta, Results on Approximation of the Rectilinear Polygon Cover Problems, Tech Rep. # CS-92-07, Department of Computer Science, Penn State, 1992.
3. P. Berman and G. Schnitger. On the Complexity of Approximating the Independent Set Problem. *Symp. on Theo. Aspects of Computing(STACS)* (1989), pp. 256-268.
4. S. Chaiken, D. J. Kleitman, M. Saks and J. Shearer. Covering Regions by Rectangles. *SIAM J. Algebraic Discrete methods*, 2 (1981), pp. 394-410.
5. H. E. Conn, and J. O'Rourke. Some restricted rectangle covering problems. Tech Rep. JHU-87/13, The Johns Hopkins University, Baltimore, MD, 1987, also appeared in *Proc. of the 1987 Allerton Conference* (1987), pp. 898-907.
6. T. H. Corman, C. E. Leiserson and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, Cambridge, Massachusetts, 1990.
7. J. Culberson and R. A. Reckhow. Covering Polygon is Hard. *Proc. 29th IEEE Symp. Found. of Comp. Sc.(FOCS)* (1988), pp. 601-611.
8. D. S. Franzblau. Performance Guarantees on a Sweep-line Heuristic for Covering Rectilinear Polygons with Rectangles. *SIAM J. Disc. Math.*, 2 (1989), pp. 307-321.
9. D. S. Franzblau and D. J. Kleitman. An Algorithm for Constructing Regions with Rectangles. *Proc. 16th ACM Symp. Theory of Comput.(STOC)*, 1984, pp. 167-174.
10. A. El Gamal. Two Dimensional Stochastic Model for Interconnections in Master Slice Integrated Circuits. *IEEE Trans. on Circuits and Systems,* CAS-28, 127-138, February, 1981.
11. A. El Gamal, J. Greene and V. P. Roychowdhury. Segmented Channel Routing is as Efficient as Conventional Routing (and Just as Hard). *Proc. 13th Conference on Advanced Research in VLSI*, UC Santa Cruz, pp. 192-211, March 1991.
12. A. El Gamal, J. Greene, J. Reyneri, E. Rogoyski, K. El-Ayat, and A. Mohsen. An Architecture for Electrically Configurable Gate Arrays. *IEEE J. Solid-State Circuits,* Vol. 24, No. 2, April, 1989, pp. 394-398.
13. M. R. Garey and D. S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1979.
14. A. Hashimoto and J. Stevens. Wire Routing by Optimizing Channel Assignment Within Large Apertures. *Proc. 8th IEEE Design Automation Workshop*, 1971.
15. H. Hsieh, et al. A Second Generation User Programmable Gate Array. *Proc. Custom Integrated Circuits Conf.*, May 1987, pp. 515-521.

16. M. Lorenzetti and D. S. Baeder. Routing. Chapter 5 in *Physical Design Automation of VLSI Systems,* B. Preas and M. Lorenzetti, eds. Benjamin/Cummings, 1988.
17. A. Lubiw. Ordering and some Combinatorial Optimization Problems with some Geometric Applications. Ph. D. Thesis, University of Waterloo, Ontario, Canada, 1985.
18. A. Lubiw. The Boolean Basis Problem and how to Cover Polygons by Rectangles. University of Waterloo, Ontario, Canada, Manuscript, 1988. Referenced in [8].
19. W. J. Masek. Some NP-complete Set Covering Problems. MIT, Cambridge, MA, unpublished manuscript. Referenced in [13].
20. C. Mead and L. Conway. *Introduction to VLSI Systems.* Addision-Wesley, Reading, MA., 1980, Chapters 2 and 4.
21. R. Motwani, A. Raghunathan and H. Saran. Perfect Graphs and Orthogonally Convex Covers. *4th SIAM Conf. on Discr. Math.* (1988).
22. T. Ohtsuki. Minimum Dissection of Rectilinear Polygons. *Proc. IEEE Symp. on Circuits and Systems* (1982), pp. 1210-1213.
23. L. Pagli, E. Lodi, F. Luccio, C. Mugnai and W. Lipski. On Two Dimensional Data Organization 2. *Fund. Inform.*, 2 (1979), pp. 211-226.
24. C. H. Papadimitrou and K. Steiglitz. Combinatorial Optimization: Algorithms and Complexity. Prentice Hall, Inc., New Jersey, 1982.
25. V. P. Roychowdhury, J. Greene and A. El Gamal. Segmented Channel Routing. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, No. 1, pp. 75-95, January 1993.
26. T. Szymanski. Dogleg Channel Routing is NP-Complete. *IEEE Trans. CAD,* CAD-4(1), pp. 31-41, Jan. 1985.