JOURNAL OF ALGORITHMS 17, 2–44 (1994)

Covering Polygons Is Hard

JOSEPH C. CULBERSON* AND ROBERT A. RECKHOW†

Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada T6G 2H1

Received February 1989; revised September 1991

We show the following for polygons without holes:

1. covering the interior or boundary of an arbitrary polygon with convex polygons is NP-hard;

2. covering the vertices of an arbitrary polygon with convex polygons is NP-complete;

3. covering the interior or boundary of an orthogonal polygon with rectangles is NP-complete.

We note that these results hold even if the polygons are required to be in general position.

1. INTRODUCTION

1.1. *Definitions*

We define a *polygon* as a closed set of points in the plane. By including the boundary in our definition of polygon, we can talk about vertex covers and boundary covers in a consistent fashion. This definition also allows for degenerate polygons consisting of one or more connected line segments or points. These will not affect the nature of our proofs.

A *convex polygon* is a finitely bounded intersection of a set of closed half planes. A polygon is thus the union of a finite collection of convex polygons. A polygon is *simple and holeless* if its boundary is a simple cycle. All polygons considered in this paper are simple and holeless unless

*Supported by Natural Sciences and Engineering Research Council Grant OGP8053. E-mail:joe @ cs.ualberta.ca.

†Research supported by a grant from the University of Alberta Central Research Fund with funds from the Natural Sciences and Engineering Research Council of Canada.

0196-6774/94 $6.00

otherwise specified. Since the boundary of a polygon is a closed Jordan curve, it divides the plane into interior and exterior regions, and consists of line segments, called *edges*, intersecting in their endpoints, called *vertices*. A polygon P is *contained* in a polygon Q if no point in P is exterior to Q. A polygon *covers* any point contained in it, and a set of points is *covered* by a set of polygons if every point in the set is covered by at least one polygon in the set of polygons. The set of convex polygons defining a simple polygon is a convex cover of the polygon. If in addition the convex polygons intersect only along their boundaries, then they form a convex *partition* of the polygon. A line is *orthogonal*, or *orthogonally aligned*, if it is parallel to one of the Cartesian axes. If all edges of a polygon are orthogonally aligned, we call it an *orthogonal polygon*; otherwise we refer to it as an *arbitrary* polygon. If no three vertices of an arbitrary polygon are in a row, we say it is in *general position*. Similarly, if no three vertices of an orthogonal polygon are in an orthogonal line then we say it is in general position.

1.2. *Previous Work*

The problems of decomposing polygons into various types of simpler polygons have a number of important practical applications [16, 25] and have received considerable attention from a theoretical perspective. O'Rourke and Supowit [25] showed that the problems of covering polygons with the minimum number of convex polygons, star-shaped polygons or spiral polygons are all NP-hard, but their constructions required the polygons to have holes. Aggarwal [1] showed that covering by star-shaped polygons is NP-hard even when the given polygon cannot contain holes.

The problem of minimally covering orthogonal polygons with rectangles was shown to be NP-complete by Masek [21] (see [13, p. 232]), but again the proof required the presence of holes. Conn and O'Rourke [6] have investigated problems of covering parts of orthogonal polygons (which may have holes) with rectangles. They showed that covering the boundary and covering the notches (i.e., the reflex angles) are NP-hard, but covering the corners (i.e., the convex vertices) could be done in polynomial time. They also showed that covering just the horizontal edges could be done in polynomial time if the polygon was in general position, but conjectured it might be NP-complete otherwise.

For orthogonal polygons, much of the work has focused on finding fast algorithms for special cases. Chaiken *et al.* [3] proved that a minimum rectangle cover was equal in size to a maximum antirectangle[1] for convex orthogonal polygons, and used this fact to obtain a polynomial time

[1]An *antirectangle* is a set of points, no two of which are contained in a common rectangle.

algorithm for this case. Franzblau and Kleitman [12] extended this result with an $O(n^2)$ algorithm for covering vertically convex orthogonal polygons. Lubiw [20] gave a polynomial time algorithm for a different subclass of orthogonal polygons known as *plaid* polygons. Franzblau [11] uses a partitioning scheme to obtain an $O(\theta \log \theta)$ approximate covering, where θ is the covering number, in $O(n \log n)$ time.

In contrast, the problem of decomposing polygons into a minimum number of nonoverlapping convex polygons has a polynomial time solution for polygons without holes [5, 4, 15, 17]. This problem becomes NP-hard when holes are allowed [18, 14]. Partitioning of orthogonal polygons into rectangles can be done in polynomial time, even when holes are allowed [19, 23]. In light of these results one might suspect that the corresponding covering problems would also be tractable for polygons without holes.

In [9], we claimed that in fact these problems are NP-hard. In this paper we present the details of our proofs. We also extend our results to show that just covering the vertices of an arbitrary polygon is NP-complete. We then show that covering the interior of an orthogonal polygon with rectangles is NP-complete, and present an outline of the construction for showing that the boundary and interior covering problems for orthogonal polygons in general are NP-complete.

In the following sections, we make liberal use of diagrams to describe the details of these constructions, first for the nonorthogonal case, and then for the case of rectangle covering of orthogonal polygons. We conclude with a discussion of some remaining open problems in the area of polygon covering.

2. Part I: Covering Arbitrary Polygons with Convex Polygons

In this section we show that the problem of minimally covering the vertices of an arbitrary simple holeless polygon Q with convex polygons interior to Q is NP-complete. The construction also shows that obtaining minimal convex covers for the boundary and the interior are NP-hard.

First, let us define the vertex covering problem as a decision problem.

Polygon Vertex Cover PVC.

Instance: A simple and holeless polygon P, and a positive integer k.

Question: Is there a covering of the vertices of P consisting of k or fewer convex polygons contained in P?

Similarly, we define the interior and boundary covering problems by

Polygon Interior Cover PIC.

Instance: A simple and holeless polygon P, and a positive integer k.

Question: Is there a covering of P consisting of k or fewer convex polygons contained in P?

Polygon Edge Cover PEC.

Instance: A simple and holeless polygon P, and a positive integer k.

Question: Is there a covering of the edges of P consisting of k or fewer convex polygons contained in P?

It is easy to see that PVC is in NP. It is only necessary to construct the *visibility graph* of the vertices of the polygon. Two vertices of P are *visible* to one another if some convex polygon contained in P covers them both. An edge exists between two vertices in the visibility graph iff they are visible to one another. A solution to PVC is represented by a clique cover of the visibility graph, and obtaining minimum clique covers is in NP. We do not know if PEC or PIC are in NP.

Our proof that PVC is NP-complete is based on a reduction from the satisfiability problem, SAT [7, 13].

Boolean Satisfiability SAT.

Instance: A set $\mathcal{U}$ of boolean variables and a collection $\mathcal{C}$ of clauses over $\mathcal{U}$.

Question: Is there a satisfying truth assignment for $\mathcal{C}$?

Let $I = (\mathcal{U}, \mathcal{C})$ be an arbitrary instance of SAT. Let $v = |\mathcal{U}|$ be the number of variables and $l = \sum_{c \in \mathcal{C}} |c|$ be the number of occurrences of literals in the clause system. Then we will show how to polynomially transform I to an instance $J = (P, k)$ of PVC, such that P can be covered by $k = 5l + 3v + 1$ convex polygons if and only if I is satisfiable.

We use the basic component design approach. These components are parts of the simple polygon P in the instance J. For each $u \in \mathcal{U}$ we construct a *variable structure*. The variable structures are the most complex part of our construction, and will be described in detail shortly. See Fig. 9. Also, for each $c \in \mathcal{C}$ we construct a *clause checker* as illustrated in Fig. 2, where there are three clause checkers. The interior of the polygon lies above these checkers and below the variable structures. See Fig. 1.

Throughout the following we will identify certain vertices or sets of vertices as being significant. We will refer to these vertices as *labeled*. Our proofs will make use of these by noting that the covering of these is critical in the sense that they will necessitate the minimum number of convex polygons required, and that these polygons are sufficient to cover all the

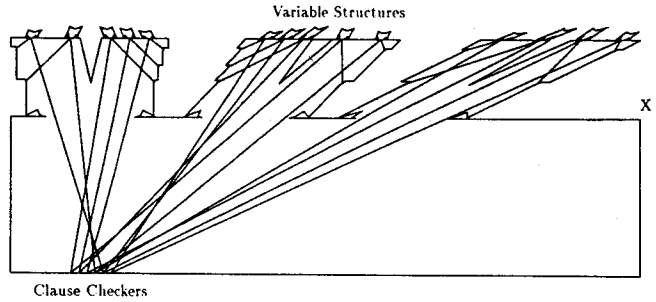


FIG. 1. A complete implementation of a simple example.

remaining vertices and indeed all of the polygon. We now outline the construction of a polygon P for an instance J, identifying properties of the construction necessary to the proof, and based on those properties prove our claim. We provide fuller details of the construction in the appendix.

One basic component used throughout the variable structures is the *beam machine* (BM). This structure is illustrated in Fig. 3. The dashed lines indicate interceptions of interior extensions of the edges, and are included to illustrate the visibility restrictions stated below. The opening at the bottom is called the *mouth* of the beam machine, and is the attachment to the rest of P. The identified vertices are part of the labeled set of vertices.

The following properties are easily verified. For purposes of this discussion, we think of the two vertices at A as one vertex, and similarly for the two vertices at B. The reason for actually having two vertices will be made clear shortly.

1. The points labeled A, B, C, D are pairwise invisible to each other, and thus at least four convex polygons are required to cover the BM.

2. All vertices in the polygon visible from C and D can be covered by two (overlapping) triangles contained in the BM. See Fig. 4.

3. The vertices labeled E and F are not visible from either C or D.

4. E and F and exactly one of A or B can be covered by a single convex polygon contained in P. This is called the *beam switch*. See Fig. 5.

FIG. 2. Clause checking system for three clauses.

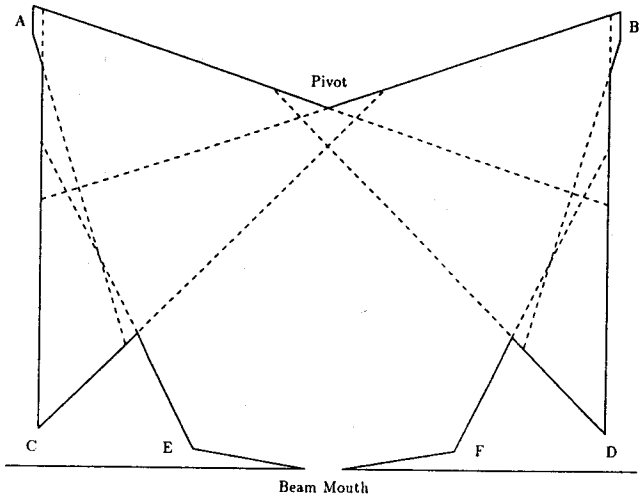


FIG. 3. Beam machine.

5. The polygons in items 2 and 4 cover all vertices in the BM except one, either B or A.

6. Any convex polygon covering either A or B and at least one of E and F and contained in P cannot extend beyond a finite distance below the beam mouth (or to either side), In particular, the remaining construc-

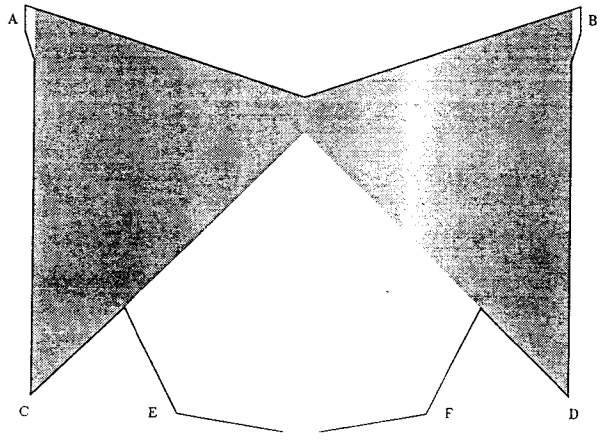


FIG. 4. Background cover for beam machine.

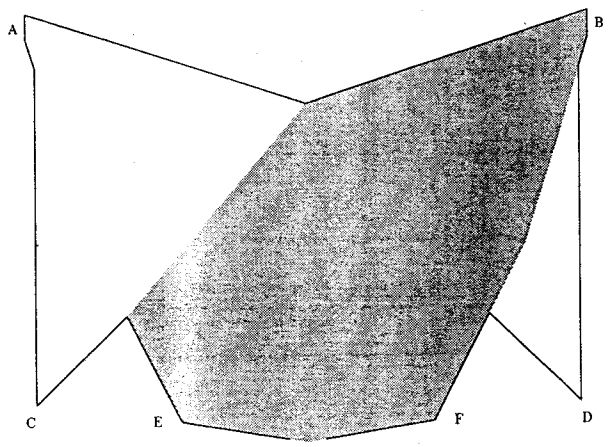


FIG. 5. Beam machine switch polygon.

tion will be such that no labeled vertex outside the BM can be covered by such a polygon.

7. A convex polygon covering either B or A can be extended indefinitely far through the beam mouth into the remainder of P. Such a polygon is called a *beam*. See Fig. 6. In particular, the remaining construction will be such that each beam will be able to cover a distinct vertex in P not in the BM.

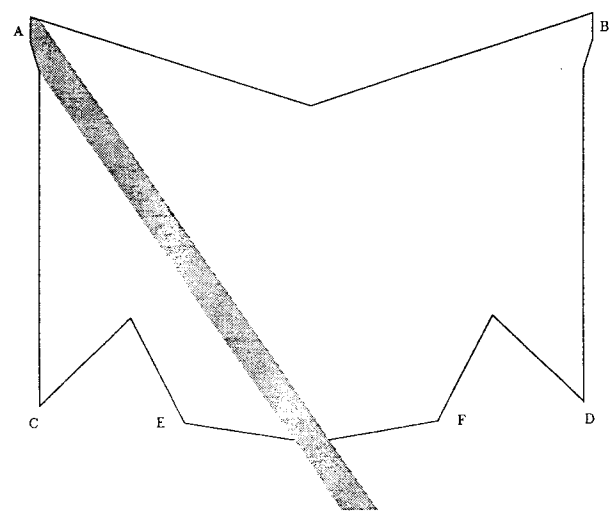


FIG. 6. Beam with parallel sides for interior cover.

These properties show that:

LEMMA 2.1. *Four convex polygons are necessary and sufficient to cove the vertices of a beam machine, and at most one of these can be extende arbitrarily far from the beam machine in the form of a beam.*

We will (geometrically) transform the beam machine so that each bean can be extended to cover one other labeled vertex. These vertices ar called the *targets*. One target will be the vertex at the apex of a claus checker. The other will be at a *beam lock* in the variable structure. Th transformations we use must not change the properties used in Lemm 2.1. The following transformations satisfy this requirement, in that they d not change visibilities within the beam machine.

Scale $Y' = sY, X' = sX$, where $0 < s$ is a scale factor to be deter mined.

Stretch $Y' = \beta Y, X' = X$, where $0 < \beta < \infty$ is a stretch factor to b determined.

Skew $Y' = Y, X' = X + \alpha Y$, where $-\infty < \alpha < \infty$ is a skewing facto to be determined.

Translate $Y' = Y + b, X' = X + a$, where $-\infty < a, b < \infty$ will b computed to place the (transformed) beam machine in its proper locatio in the variable structure under construction.

See Figs. 7 and 8 for examples of how skew and stretch affect the bean angles. Note in particular that neither transformation changes the bean mouth width. At this point we explain why we need two vertices at A (an B). If we are covering the interior, then the sides of the beam from A ar parallel, since the entire region at A (Fig. 4) must be covered. This mean that the horizontal width of the beam is not affected by either skew o stretch, and thus the beam may still cover a clause checker (or beam lock after these transformations with no further changes. For vertex cover however, only the vertices at A or B need to be covered, and so keepin the mouth width fixed, the transformations will cause the angle betwee the sides of the beam to change, and thus it may cover more than on clause checker. By moving the vertices at A sufficiently close to th remainder of the beam machine after transformation, we can focus th beams so that the sides are arbitrarily close to parallel. The beam will no just cover the mouth of a clause checker, even for the vertex covering case and yet still yield a covering that can be extended to cover the boundar and the interior. This simple device makes vertex cover a special case o interior or boundary cover, and so simplifies the remainder of our proof The actual locations of these vertices are computed after the othe transformations have been applied.

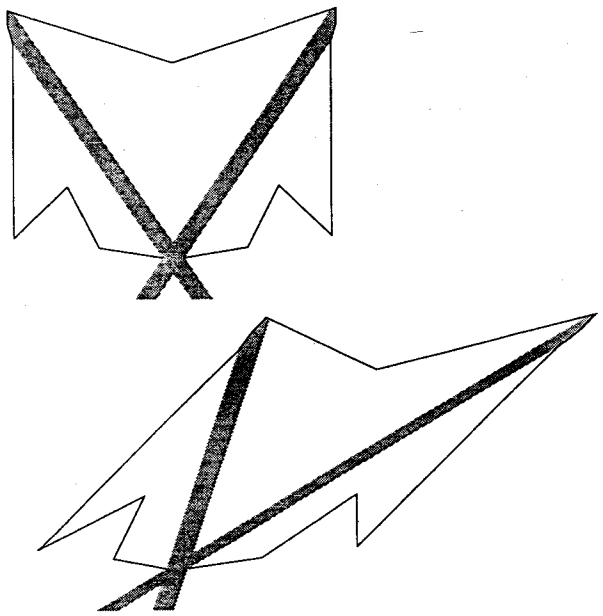


FIG. 7. The effect on beam angles of skewing with $\alpha = 1.0$.

LEMMA 2.2. *The transformations scale, stretch, skew, and translate do not change the visibilities between vertices within a polygon.*

Proof. The proof follows from observing that these are linear transformations in which straight lines map to straight lines preserving intersections. □

Note that by joining the ends of the beam mouth, the beam machine forms a simple polygon, and that all properties necessary to Lemma 2.1 hold within that polygon. Also, note that under each of the transformations, parallel lines map to parallel lines. These properties plus iteration on the previous lemma prove

LEMMA 2.3. *Four convex polygons are necessary and sufficient to cover a beam machine to which any set of the above transformations has been applied, and exactly one of the four polygons can be extended indefinitely far through the beam mouth.*

We now outline the construction of a variable structure. For the purposes of this construction, we assume a local origin at the center of the variable mouth (see Fig. 9) and we place the variable structure directly above the clause checkers. After construction, the variable structure will

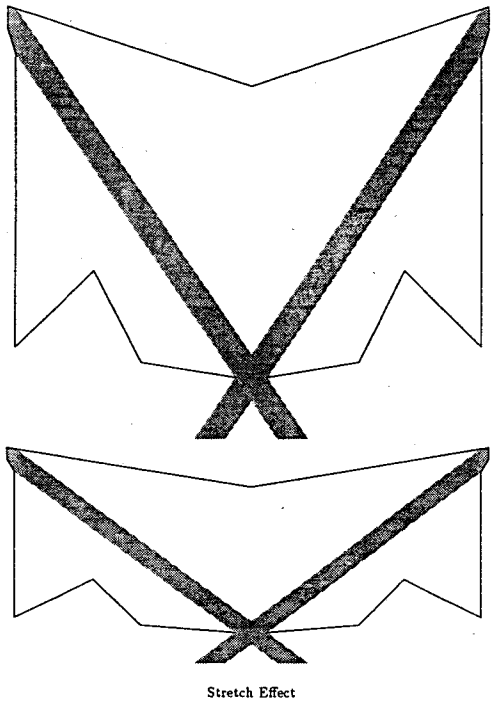


FIG. 8. The effect on beam angles of stretch with $\beta = 0.5$.

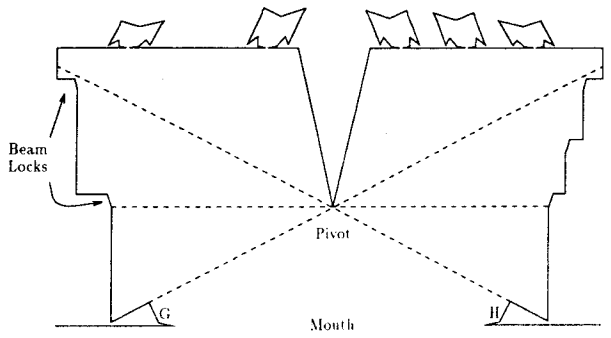


FIG. 9. Variable structure.

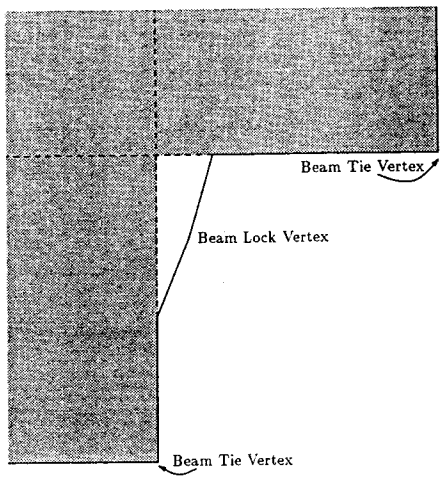


FIG. 10. Details of a beam lock.

be moved aside to allow the construction of another variable structure by applying a skewing operation.

For each occurrence of a positive literal in some clause, we place a beam machine at the top of the variable structure on the right side and a corresponding *beam lock* on the right side of the variable structure. See Figs. 9 and 10. The beam locks are joined by orthogonal edges. These edges intersect at vertices called *beam ties*. In a similar manner, for each occurrence as a negated literal, we place a beam machine on the top left of the variable structure and a corresponding beam lock on the left side. The beam lock and beam tie vertices are labeled.

The beam lock vertices on one side of the variable structure are all visible to one another, and to the vertices labeled G and H near the mouth of the variable. The beam lock vertices on one side of the variable will not be visible to those on the other side. None will be visible to any other labeled vertex except the ones in the corresponding beam machine (i.e., A for beam locks on the right, and B for beam locks on the left). The beam tie vertices are visible to no other labeled vertex, and thus each one requires a separate covering polygon. See Fig. 11. We require at least one additional polygon to cover the vertices labeled G and H at the mouth of the variable structure, and this can also cover the beam locks on just one side of the variable structure. We refer to those polygon as the *variable switch*. See Fig. 12.

The construction is completed by creating one variable structure for each variable, and skewing each one so that they do not overlap. The skew

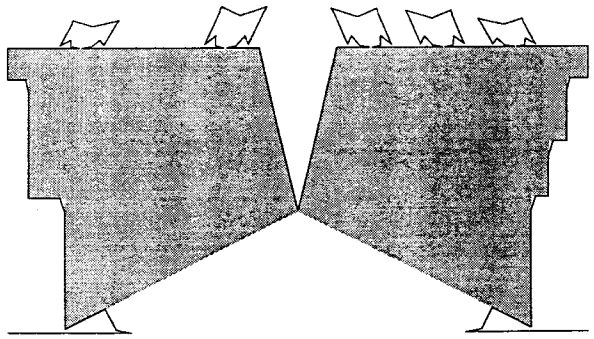


FIG. 11. Background cover for variable structure.

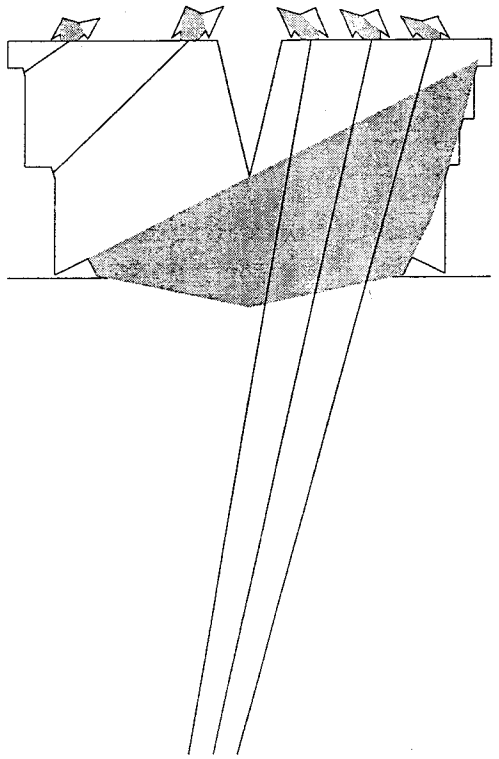


FIG. 12. Switch polygon for variable structure.

assumes the x-axis at the mouth line of the clause checkers, and since a beam covers an entire clause checker, skewing will not affect the visibility of the target vertices, provided that the slope of the right side of the each clause checker is shallow enough. The variables and clause checkers are then connected to complete a polygon, as illustrated in Fig. 1. The depth of the clause checkers is adjusted so that the extensions of the right sides intersect the top of the clause region between the rightmost variable mouth and the point labeled X. Thus, X is not visible from any clause checking vertex.

Based on the properties claimed in the preceding paragraphs, we claim

THEOREM 2.1. *PVC is NP-complete.*

Proof. We have already shown that PVC is in NP.

We show that the instance of PVC constructed can be covered using $k = 5l + 3v + 1$ convex polygons iff the instance of SAT is satisfiable. First, note that the setting of the variable switches is trivially mapped to corresponding truth values of the variables, by letting the case of the switch covering the right beam locks correspond to a setting of true for the corresponding variable. Suppose that the instance of SAT is true. Then take any satisfying truth assignment and set the corresponding switches in the variable structures. Since every clause has at least one true literal, and under the switch setting the corresponding beam machine can send a beam to cover the clause checking device, every clause checker will be covered by a beam. We note that for every literal, there is a beam machine and one beam tie. The beam machine requires four convex polygons and the beam tie one for a total of $5l$ polygons. In addition, there is one additional beam tie vertex on each side of each variable structure, and one switch in each variable, for a total of $3v$ polygons. Finally, we need one polygon to cover the vertex X in Fig. 1, which covers the vertices between the variable structures and the clause checkers. The total as claimed is $5l + 3v + 1$.

Conversely, suppose that the instance of PVC is true. As computed in the previous paragraph, we need at least $5l + 3v$ polygons to cover the labeled vertices in the variable structures. We also need one to cover X, which is not visible to any of the clause checker vertices. Thus, if the clause checkers are to be covered, it must be by extension of some of the polygons covering the variable structures; but the only labeled vertices visible to the clause checkers are the ones in the corresponding beam machines, i.e., the beams. If these beams are used to cover the clause checkers, then the corresponding beam locks must be covered by another polygon, and the only polygon available is the variable switch, since no other labeled vertex in the variable structure can see the beam locks except G and H. That is, the beam locks that are uncovered by beams,

and the mouth vertices of the variable structure must be covered by a single polygon; but this can only happen if the beam locks in each variable that are not covered by beams are all on one side of the variable, and this constitutes a setting of the switches, which defines a truth assignment.

Finally, we claim that the construction can be done in polynomial time, using rational arithmetic with $O(\log n)$ bits. Scaling the beam machines (beam locks and clause checkers) by one-half doubles the number that will fit in the variable structure. Doubling the width doubles the number of variable structures. Either operation requires only one extra bit of accuracy. In the following section, we specify the basic components using rational coordinates, and only the basic arithmetic operations are required in our construction.

In fact, the construction only requires a linear number of arithmetic operations. This follows since each transformation is applied once to the vertices of each beam machine, followed by one skew transformation to the entire variable. The number of vertices is clearly linear in the size of the instance of SAT. □

We have been careful in our construction that a vertex covering can be extended to an interior or boundary covering by enlarging the polygons so that no new polygons are required.

This immediately gives us

COROLLARY 2.1. *PEC and PIC are NP-hard.*

Certain vertex alignments are evident in our construction; namely the mouths of the beam machines are in line as are the mouths of the variables and the clauses. These alignments are not necessary to our construction, since we only require in each case that the vertices should be vertices on some convex polygon. This can be obtained by perturbing the vertices by tiny amounts. This would show that our results hold even for general position polygons, but we leave the details to the interested reader.

We note that Shermer [27] has constructed an independent proof that interior covering is NP-hard based on a reduction from Exact Cover by 3-Sets.

2.1. *Appendix I: Construction Details*

We now describe in somewhat more detail the construction of an instance of PVC from an instance of SAT. These details should be enough to allow the skeptical reader to verify the properties claimed in the previous section. In generating the examples, we found that quite a few

constraints had to be met, not the least of which was obtaining coordinates which looked nice when drawn. Thus, we list the major coordinates as an example of how the properties can be met, although there are of course infinitely many other satisfying constructions.

The Cartesian coordinates of the right hand side of the beam machine, with local origin centered in the mouth, are, in counterclockwise order from the right side of the mouth.

$$(3,0), (21,3), (30,21), (45,6), (45,63), (93/2,68)^*,$$
$$(93/2,145/2)^*, (0,57).$$

The left part of the beam machine is obtained by negating the x-coordinates. The two vertices marked by $*$ will in fact be moved to focus the beams for the vertex cover as described in the previous section. We refer to these as the *ear vertices*. For interior or boundary cover, this adjustment is not needed.

For this set of coordinates we can see that the slopes of the beams (i.e., assuming the sides are parallel, and before transformation) are $\pm 3/2$. The parameters for the transformations can be computed as follows. If the center of the mouth of the beam machine is to be located at (x_1, y_1), the target vertex for the left beam (that is, the beam of positive slope) is (x_2, y_2) and the target for the right beam is (x_3, y_3), then the parameters for the transformations are

$$\alpha = \frac{1}{2}\left(\frac{x_2 - x_1}{y_2 - y_1} + \frac{x_3 - x_1}{y_3 - y_1}\right)$$

$$\beta = -2\Big/\left(3*\left(\frac{x_3 - x_1}{y_3 - y_1} - \alpha\right)\right)$$

$$a = x_1, \qquad b = y_1.$$

Our construction ensures that the targets are always below the beam mouth, and so we do not have to worry about division by zero. We leave the verification of these formulas to the reader.

The scale factor s will be the same for all beam machines, and will depend upon the number of occurrences of any given literal, and the number of clauses.

The vertices defining the variable structures mouth are at $(-175, 0)$ and $(175, 0)$ (assuming local origin at the center of the mouth). Proceeding in a counterclockwise direction from the right side of the variable mouth, the next two vertices are at $(189, 3)$ and $(200, 25)$. The left side is symmetric.

The *pivot vertex*, the one directly above the center of the mouth is locate at $(0, 128)$ and the one adjacent to it on the top right is at $(40, 300)$, whil the far top right is $(300, 300)$. The beam machine mouths are interpolate over the range $(75, 300)$ to $(225, 300)$.

The beam lock vertices must be located on a convex polygonal chain, s that the variable switch may cover them. They must also be located abov the pivot vertex, to prevent them from being visible to the beam locks o the other side of the variable structure. To accomplish this, the x-coordi nate is interpolated over the range $244 \cdots 280$, and the correspondin y-coordinate is computed by the quadratic $x^2/72 - 34x/9 + 2060/9$ The sides of the beam lock are then computed by taking lines from th beam lock to points interpolated between the lock vertices, and truncatin these lines so that the parallel beam covers the entire lock. This require computing a line intersection, where the equation for the lines definin the beam are easily determined since we know the location of the beam machine and the lock vertex. The end points of these segments are then joined by the orthogonal lines to the next beam lock.

Using the above, the following algorithm performs the transformation.

1. Count the number of clauses, and the maximum number of occurrences of any literal. The maximum m_c of these two determines the scale s, by $s = \min\{s', 3s'/m_c\}$, where s' is the scale used in the diagrams, and is thus suitable for up to three literals or six clauses. This is the scale factor used for transforming beam machines, and to compute clause checker width (i.e., multiply by the width of the clauses in the diagram).
2. Interpolate the mouth centers of the clause checkers along the line $(-75, 0)$ to $(75, 0)$. Compute the depth d of the clause checkers, so that the right slope from the leftmost clause will intersect the point $(1 + 600n, 600)$, where 600 is the overall width of a variable structure, and n is the number of variables.
3. For each variable $1 \le k \le n$, do the following
   (a) Translate the variable so that the mouth center is at $(0, 600)$. By placing it one variable width above the clause checkers, each unit of skew moves it one variable width to the right.
   (b) Interpolate the beam locks as described above.
   (c) Interpolate and transform the beam machine so that one beam covers the corresponding clause and the other the corresponding beam lock. After placement, compute the location of the ear vertices to focus the beams on the targets. This is done by computing the intersection of two lines for each vertex, e.g., the line joining the end of the clause checker and the same side of

the beam mouth, and the line joining the ear vertex to the beam machine.

(d) Skew the entire variable by $\alpha = k$.

4. Join the variables and complete the box joining them to the clause checkers.

3. Part II: Covering Orthogonal Polygons by Rectangles

3.1. *Interior Coverings*

An *orthogonal polygon* is a polygon in which every edge is aligned with one of the axes; that is, either vertical or horizontal. A *rectangle covering* is a covering by rectangles which are also orthogonally aligned.

We define the rectangle covering problem as a decision problem.

Rectangle Cover RC.

Instance: A simple and holeless orthogonal polygon P, and a positive integer k.

Question: Is there a covering of P consisting of k or fewer rectangles contained in P?

We will use a reduction from 3SAT [13].

Three Satisfiability 3SAT.

Instance: A collection $C = \{c_1, c_2, \ldots, c_m\}$ of clauses on a finite set U of variables such that $|c_i| = 3$ for $1 \leq i \leq m$.

Question: Is there a truth assignment for U that satisfies all the clauses in C?

As for general polygons in Part I, we use a component design to construct a polygon such that a covering of given size may be achieved if and only if the instance of 3SAT is satisfiable. The construction is somewhat similar in that we use beam machines to deliver signals from variable structures to clause checkers. Indeed, the concept of rectangle beam machines was a direct result of the discovery and use of beam machines for the general case.

There are some obvious differences. The most notable result of restricting the covering to orthogonal rectangles is that the beams must be orthogonally aligned and cannot be aimed in arbitrary directions. This fact makes the construction somewhat more complex than that of the previous section, in that we must find a way to direct a beam from an arbitrary

variable structure to an arbitrary clause checker. The beam machines serv handily here in that they act as effective relay devices.

During our research into this problem, the invention of our devices an their connections was based on definitions of dent diagrams and sourc graphs for rectangle coverings [10], similar to those used for our results i orthogonally convex coverings [8]. Indeed, it was a failed attempt to exten the dent diagram approach to arbitrary polygons that led to the results i this paper [10], as one of our counter examples led directly to the bear machine for arbitrary polygons.

In the following presentations, we identify sets of small subregion which must be covered and for which any covering can be extended t cover the entire polygon without adding additional rectangles. Thes regions are sources in terms of the definitions of [10], but for the purpose of this paper, the simplified definitions below are sufficient.

In this section we generally outline how each device is constructed as w describe the device. This is necessary as determining the number o rectangles required to cover a device (and the entire polygon) usuall depends on understanding the construction. In order not to be tedious our descriptions will sometimes sacrifice formality in favor of brevity an understandability.

If we extend the edges of any orthogonal polygon internally until the intersect the boundary then the polygon will be divided into $O(n^2$ rectangles, called *cells*.[2] We will identify certain critical cells, just as w labeled certain critical vertices in the previous section. We call thes identified cells *sources*.

The following observations has important consequences.

Observation. Any rectangle covering a portion of a cell can be ex tended to cover the entire cell.

This follows since the rectangle can be extended along any axis until i encounters an edge of the polygon, and the edges of the cells are define by extensions to the edges of the polygon.

A rectangle is *maximal* it it is not properly contained in any othe rectangle contained in the polygon. In determining a minimum covering we need only consider coverings consisting of maximal rectangles. Cou pling this fact with the observation, it follows that the desired covering can be defined in terms of subsets of the cells of the polygon, and thus the rectangle covering problem is in NP.

[2] Precisely speaking, using our closed definitions of polygons, the cells are the open region between (the extensions of) the edges. The following observation holds for rectangle covering portions of these open regions.

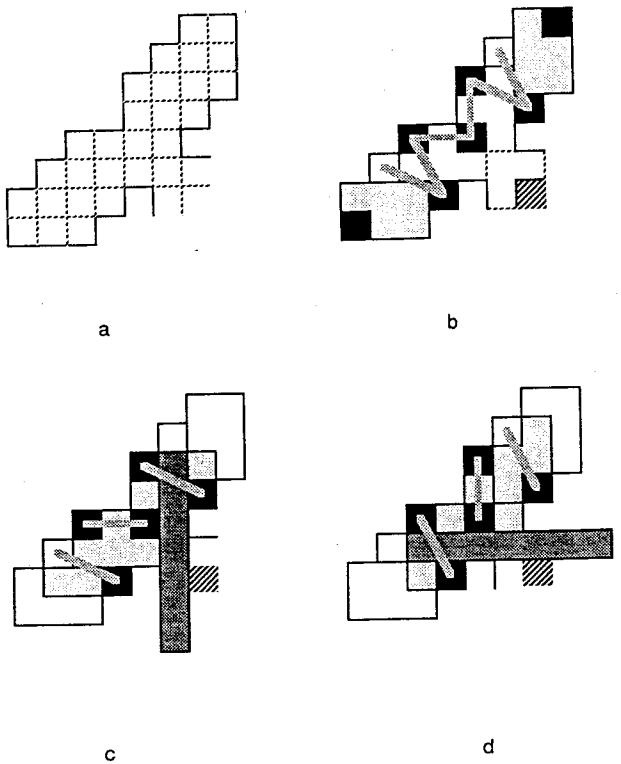


FIG. 13. Rectangle beam machine (see text).

If two cells can be covered by a single rectangle, then we say they are *visible* to one another or that they *see* one another. The observation implies that there is no case of partial visibility between cells, and thus visibility is a well defined relation on cells. We say that a set of cells (or sources) is *independent* if no two are pairwise visible.[3]

The first structure we will examine is the *beam machine*, which is illustrated in Fig. 13. In Fig. 13a we show the beam machine with dashed lines indicating the cells within the beam machine. In part (b) we identify the sources which are crucial to its operation, by small black or white squares internal to the beam machine. In Fig. 13b the region with the dashed outline indicates the *mouth* of the beam machine, and the small striped square is the *mouth source*. When speaking of covering the beam machine, we mean covering the interior of the beam machine and not the

[3]An independent set forms an *anti-rectangle* in the terminology of [3].

mouth. Covering the mouth, and in particular the mouth source, will be described later when we discuss background covers of composite devices.

In general, we will refer to sources by the shading used to identify them in the diagrams. It is easily seen that no black source is visible to any cell outside the mouth of the beam machine. Certain pairs of cells are visible internally in the beam machine as indicated by the gray polygonal line. Note that the two extreme sources cannot see any other source, and thus require separate covering polygons. Unique maximal covers for these sources are shown in gray in Fig. 13b.

The remaining sources can be covered by four rectangles in two ways such that one rectangle, covering only a white source, may be extended through the mouth of the beam machine an indefinite distance, as illustrated in Figs. 13c and 13d. The *positions* of these latter rectangles are called *beams*. We say a beam is *on* if the covering of the beam machine includes the rectangle covering the beam. Otherwise the beam is *off*. We may occasionally abuse our terminology by using the term "beam" to refer to the rectangle covering a beam that is on. Beams are generally shared between two beam machines, or between a beam machine and another device, as illustrated, for example, in Fig. 18.

Additional coverings of six rectangles exist in which no rectangle may be extended out of the beam machine (that is, both beams are off) but these have no bearing on the proof.

LEMMA 3.1. *Six rectangles are necessary and sufficient to cover a beam machine, and exactly one of these may be extended through the beam mouth. The extension may be either horizontal or vertical but not both.*

Proof. Sufficiency is established in the preceding discussion. To see that six rectangles are required, we observe that there are six independent sources on the path from the leftmost edge to the topmost edge.

To see that at most one of the six rectangles can be extended beyond the beam mouth, we note first that only a rectangle covering a single white source may be so extended. The sources at the extremes of the beam machine require one rectangle each. This leaves four rectangles to cover the remaining seven sources, and since a single rectangle may cover at most two of these sources, this implies that three of the four must cover a pair of sources. □

Although six polygons are necessary and sufficient for a single beam machine, we need to be careful when coupling two beam machines or other devices not to double count the beams. For purposes of counting, it is preferable to separate the interior cover of the beam machine from the beams.

LEMMA 3.2. *Five rectangles are necessary and sufficient to cover the interior of a beam machine, if and only if at least one of the incident beams is on.*

The five rectangles are the *background* cover of the interior of the beam machine. In each of our devices, we will make the distinction between covering of the background and covering the beams. When counting the required number of rectangles, separate counts will be made for the beams.

In Part I of this paper, we can interpret either side of the variable structure as a device that uses the beam machines to fanout the signal generated by the variable switch. The beams then transmit the signals to the clause checking devices. Similarly, we need a fanout device for the rectangle covering problem. Such a device, which we call a *beam splitter*, is illustrated in Fig. 14. We prefer to think of the striped beam at the bottom of Fig. 14a as the input signal, and the three (in this case) gray beams at the left as the output signals. The number of output beams is the *fanout* of the beam splitter.

Figure 14a illustrates the beam structure of a splitter with fanout 3, while Fig. 14b illustrates the rectangles required to cover a splitter with fanout 2 in addition to those required to cover the beam machines. We refer to these rectangles as the *background* covering of the device. These leave two white sources uncovered, which we call *beam locks*. These function in a manner similar to the beam locks of Part I. As illustrated in Fig. 14a, all the locks may be covered by a single gray beam, or they may be covered individually by the striped beams. In the first instance, the output beams may all be on. Otherwise to obtain a minimum covering the internal striped beams must *all* be on.

We illustrate the method of construction of a splitter with fanout $f \geq 1$ in Fig. 14c. For each output beam $f > 1$ desired, we insert an additional step in the staircase at A, extending the edge at C to compensate; extend the side B and insert a beam lock; and insert a beam machine at D to align with the beam lock. Clearly, this construction requires linear time in f.

Based on this construction, we obtain the following:

LEMMA 3.3. *A beam splitter with fanout f requires* $7f + 6$ *background rectangles and* $f + 1$ *beam rectangles to minimally cover it, and requires* $O(f)$ *time to construct.*

Proof. The construction shows that we require a rectangle at A and one at B for each output beam, plus one for the uppermost black source in Fig. 14b, for a total of $2f + 1$ background rectangles. We will be careful in further construction that no other source is visible from a source at A. Clearly, the other sources cannot be seen from outside the splitter. Using

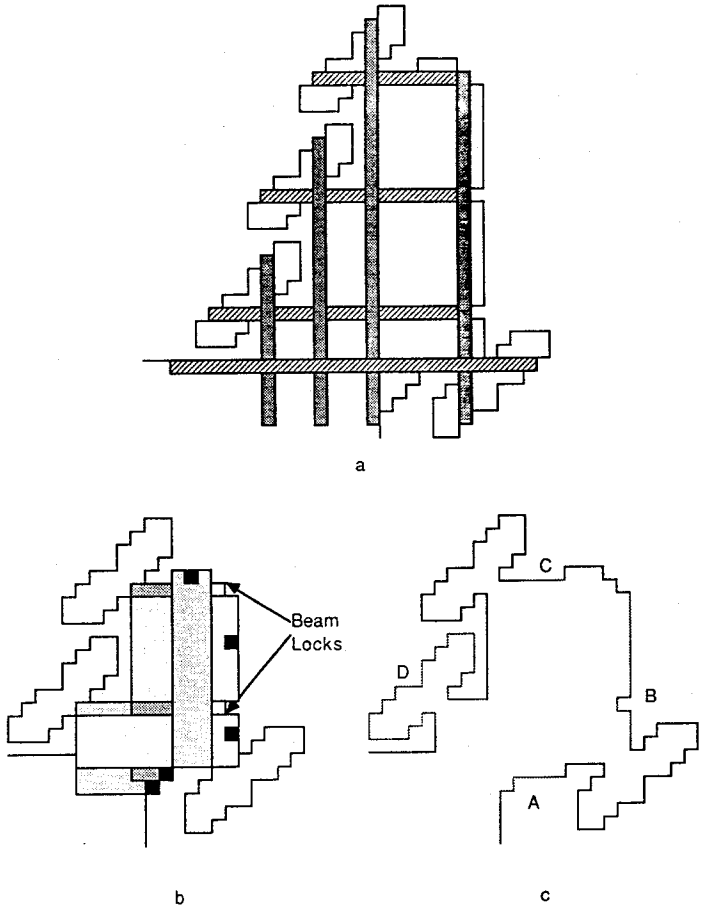


FIG. 14. Beam splitter (see text).

Lemma 3.2, $5(f + 1)$ background rectangles are required for the interiors of the beam machines. Finally, for a minimum cover, at least one beam must be on in each beam machine, and there are $f + 1$ beam machines. In addition, the beam locks must be covered and these can only be covered by beams unless extra rectangles are created. □

Note that we counted all beams as part of the beam splitting device. We will be careful not to count these beams again as part of other devices.

The next device we need is an inverter, as illustrated in Fig. 15. At the same time we will illustrate the 3-clause checker. The inverter could act as a 2-clause checker if we desired, and it is easy to see that the construction

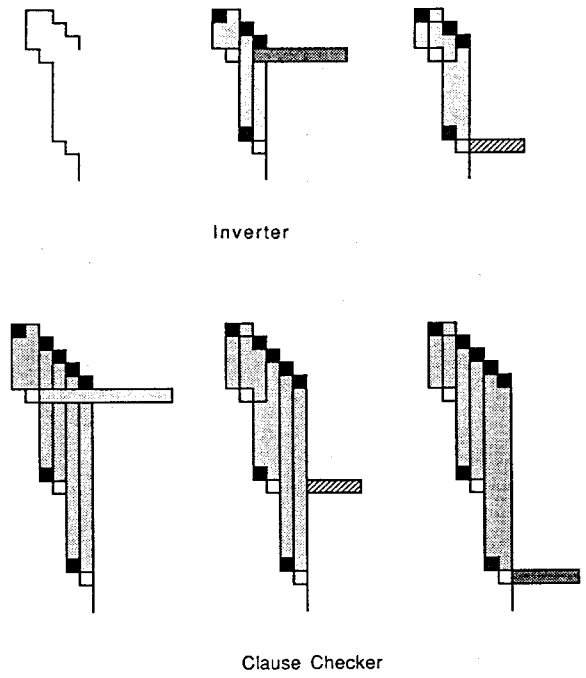


FIG. 15. Inverter and clause checker.

could be extended to clause checkers of any cardinality. In fact, this is the only change necessary to make the reduction from SAT instead of 3SAT.

Of the three independent black sources at the top of the inverter, one is not visible to any other source. The white sources are located on beams as indicated. We will ensure that the remaining nonwhite source is not visible to any source outside the inverter. Thus,

LEMMA 3.4. *An inverter may be covered with three rectangles if and only if at least one of the two incident beams is on.*

Similarly:

LEMMA 3.5. *A clause checker may be covered with five rectangles if and only if at least one of the three incident beams is on.*

Note that we do not count the beams as part of the cover for either of these structures.

We are now ready to construct a variable structure, analogous to that of Part I, which is easily understood given the existence of inverters and beam splitters, despite the rather complex appearance of Fig. 16. Suppose

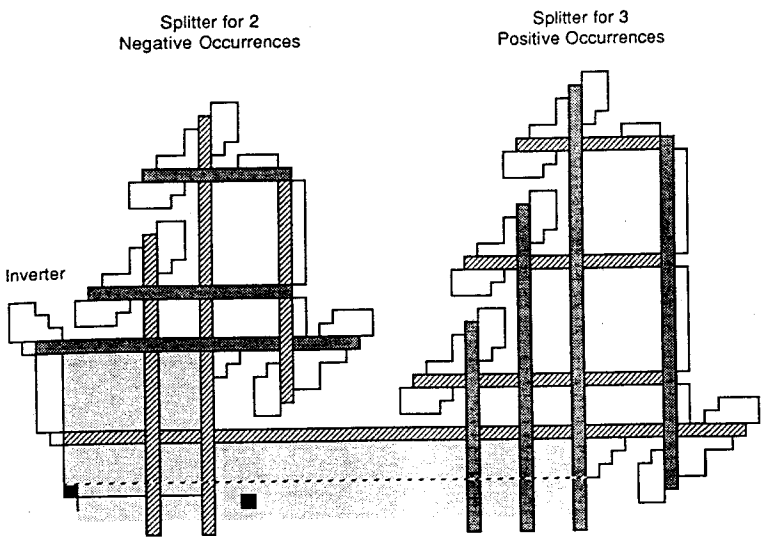


FIG. 16. Rectangle variable structure.

the variable we wish to represent has f_1 negative occurrences and f_2 positive occurrences.[4] Then we simply construct two splitters and link them to an inverter on the far left, as shown in Fig. 16. The reader can verify that the black source in the inverter is not visible to any source outside the inverter and that the white sources are visible to the sources in the beam machines that are aimed at them from the splitters. To complete the variable structure, we make a notch to create the black source at the lower left, so that it is not visible to the sources near the mouth of the splitter on the right. The remaining black square indicates that there is a source in the mouth of the variable structure that also needs to be covered, although the source will in general be much larger than the square.

LEMMA 3.6. *The variable structure requires two background rectangles in addition to those required to cover the inverter and the beam splitters. It takes $O(1)$ time to construct in addition to the time used to construct the splitters.*

It is now easy to construct any number of variable structures and line them up side by side, joined at their bases, as in Fig. 17. In that figure there are two variable structures; the left one has two occurrences as a

[4] We make the usual assumption that $f_1, f_2 \geq 1$.

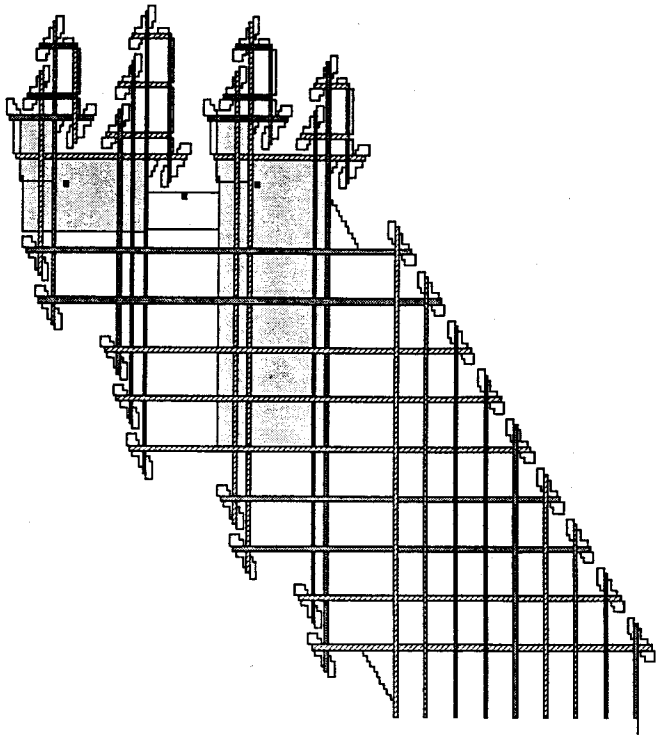


FIG. 17. Two variable structures and initial joint.

negated literal, and three as a positive literal. All beams exiting the variable structures will be parallel (and vertically aligned).

The structure below the variables in Fig. 17 we call a *joint* and it is the basis for propagating the signals from the variable structures to the clause checkers. It serves as the basis for the construction of the line switch to be described later. The clause checkers will be attached after some further insertions of line switches, as shown in Fig. 18. Note that a joint consists of two diagonal rows of beam machines aligned so that the horizontal beams are shared, with one such pair for each beam exiting a variable structure. The idea is that for each occurrence of a literal there will be a sequence of beams from a beam lock in some variable structure to the appropriate clause checker. We call such a sequence a *beam line*.

We call the vertical beams at the top of the joint structure the *input* beams, and the vertical beams at the bottom the *output* beams. The horizontal beam are the *joining* beams.

The background for a joint (for six beam lines) is illustrated in Fig. 19. The staircases at the top and bottom of the figure ensure that there is one

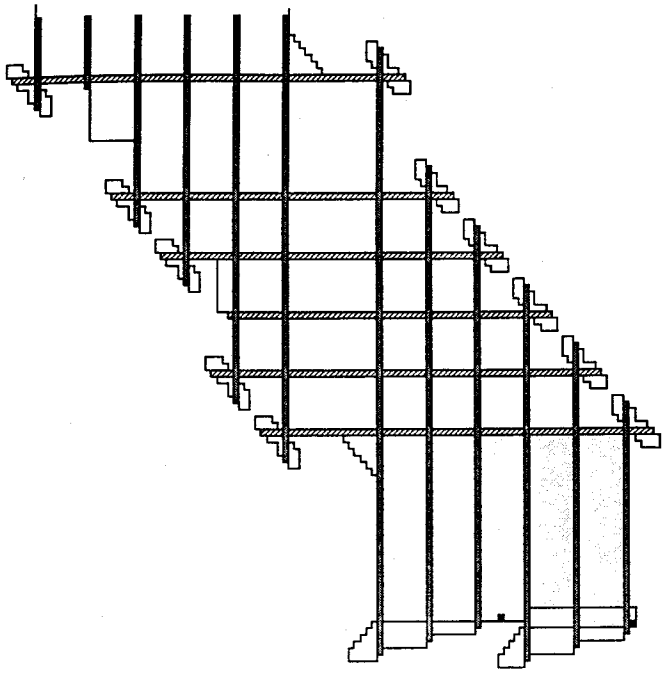


FIG. 18. Line switch and two attached clauses.

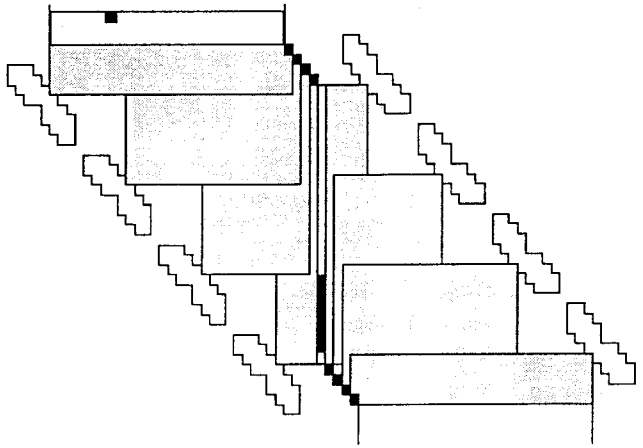


FIG. 19. Background cover for a joint.

rectangle required for each beam machine, and as shown these are sufficient to cover all of the mouth sources of the beam machines. In addition, one rectangle will be required to cover the long black source, and one to cover the top most source, indicated by the small black square at the top of the diagram. (In general, the source will be larger than this.) In Fig. 17 this rectangle covers the region between the two variable mouths. We refer to this as the *joining rectangle*. The uncovered region at the bottom of Fig. 19 will be covered by the joining rectangle of the line switch attached to it and to be described next.

From this discussion, we have

LEMMA 3.7. *If there are l beam lines, then $12l + 2$ background rectangles, and l beams are necessary and sufficient to cover a joint.*

Proof. The background count for the joint is $2l + 2$ for the rectangles not in beam machines, and $10l$ for the rectangles in the $2l$ beam machines. If the joining beam of every pair of beam machines is on, then the device is covered, and thus l beams are required. If an input beam is on, then it is still necessary to have either the joining beam or the output beam in the corresponding line on to cover the device. (Recall that the input beams are not counted as part of the joint as they are already counted as part of the splitters in the variable structures.) □

We want these beam lines to enter different clauses, as at the bottom of Fig. 18, which has two clauses checkers. Thus, we will need to switch the order of the beam lines before attaching the clause checkers. Figs. 20 and 21 illustrate how this may be accomplished, using a device called the *line switch*. This device is similar to the joint, except that one input beam (the second from the left in the example in Fig. 20) does not align with a beam machine, but rather aligns with a small source in the polygon. See the cell in the upper left circle in Fig. 21. Also, one joining beam also does not align with a beam machine on the left, but rather with a different small source. See the lower circle in Fig. 21, which corresponds to the left end of the third beam from the bottom in Fig. 20. We call these the *switch beams*. In Fig. 21 we see that as in the joint, we need $2l + 2$ rectangles outside the beam machines, one for each black source in the figure. One of these rectangles is not shown, the one covering the black source in the top right circled region. This rectangle may be used to cover either one of the circled white sources but not both. The other one must be covered by a beam.

LEMMA 3.8. *If there are l beam lines, then $12l - 3$ background rectangles and l beam rectangles are necessary and sufficient to cover a line switch.*

Proof. As Lemma 3.7, there are $12l + 2$ independent black sources not in the beam machines, and there are $2l - 1$ beam machines which by

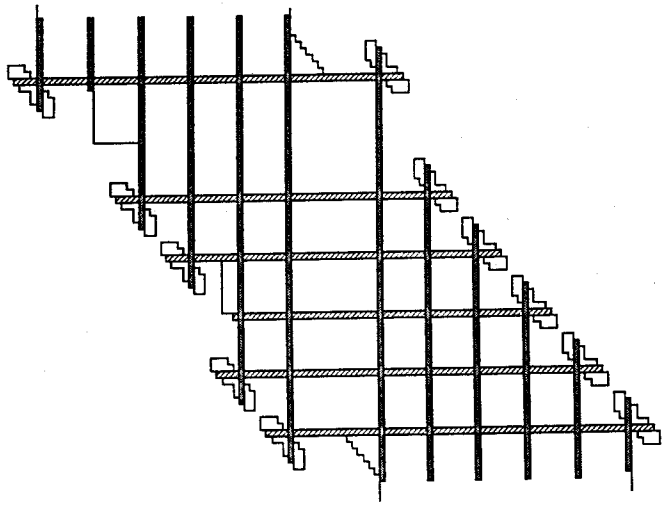


FIG. 20. Line switch.

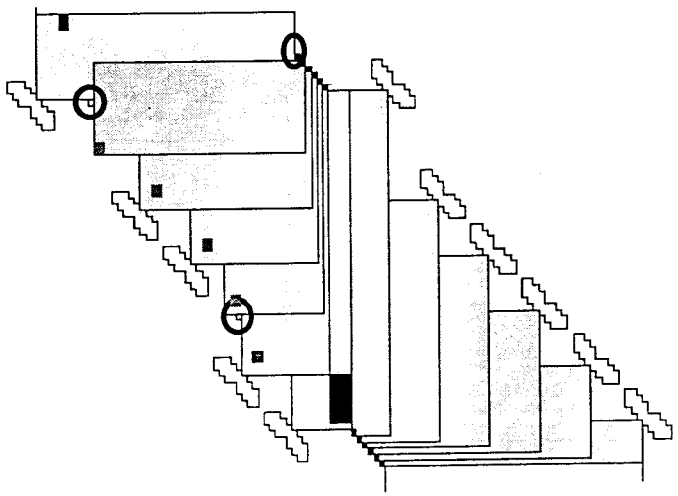


FIG. 21. Background cover for line switch.

Lemma 3.2 require five rectangles each. As illustrated, these are sufficient to cover all but one of the white sources in Fig. 21. On the other hand, using this number of rectangles, it is not possible to cover both white sources without leaving one of the gray sources in Fig. 21 uncovered. There are l beam machines on the right side of the line switch, and thus l beam rectangles are required. It is always possible to use the joining beam to cover one of the white sources, so the count is also sufficient. □

Note that it is important that no source outside the line switch be visible to the white or gray sources in the line switch. To ensure that we do not allow, for example, a variable mouth cover to cover one of these, we insist that a joint be the first device attached to the variable structures.

In the general case, a line switch is constructed as follows. If we label the input beams from left to right as $b_1, \ldots, b_i, \ldots, b_{j-1}, b_j, b_{j+1}, \ldots, b_l$ and we wish the output beam lines in the order $b_1, \ldots, b_{j-1}, b_j, b_i, b_{j+1}, \ldots, b_l$, then we start with a joint, and replace the beam machine at the end of the ith input beam by a pair of notches as illustrated for $i = 2$ in Fig. 20. We shift the right side down an appropriate amount so no beam machine will be in the location of the missing joining beam. We then shift the left side down after the jth beam machine to allow the pair of notches defining the joining switch beam. Clearly, this construction takes $O(l)$ time and can be used to switch any beam line i to a position j, where $1 \leq i < j \leq l$.

LEMMA 3.9. *A sequence of at most $l - 1$ line switches can be constructed to sort l beam lines into any output order.*

Proof. Using the above construction, on the kth line switch, $1 \leq k < l$, move the $l - k + 1$st line to its correct position. If it is already in its correct position, then we don't need the kth switch. This amounts to a selection sort, where we select the lines in order from right to left. □

Finally, we must connect the clause checkers to the last line switch, as shown in Fig. 18. This takes $O(l)$ time, since we have $O(l)$ clause checkers. The following is trivial upon examining Fig. 18.

LEMMA 3.10. *One background rectangle per clause is necessary and sufficient to cover the piece joining the clauses and the line switch.*

We are now ready for our main theorem.

THEOREM 3.1. *RC is NP-complete.*

Proof. We have already shown that the problem is in NP.

We now show that for any instance I of 3SAT, we can obtain an instance $J = (P, k)$ of RC using the construction outlined above where P is an orthogonal polygon. Let $l = \sum_{i=1}^{m} |c_i|$ be the number of occurrences of literals in the instance I. Let v be the number of variables, and c be the number of clauses. Note that $l = 3c$. Let s be the number of line switches used in the construction, where by Lemma 3.9, $s < l$. Then we show that the constructed polygon P can be covered by $6c + 21l + 2 + s(13l - 3) + 19v$ rectangles if and only if I is satisfiable.

To obtain this number, we total up over the various components. Firs we total the rectangles required for a variable, keeping separate the bean rectangles for the time being. We assume that every inverter has at leas one of the incident beams on for this count, then show later that this ca happen with the allowed number of beam rectangles if and only if I i satisfiable.

First we total the background rectangles. Each variable has one inverte which requires three rectangles, and two mouth rectangles, plus two bear splitters. Totaling, we have $5v$ plus the number of background rectangle in the beam splitters. We note that the sum of all the fanouts in th splitters is l, and, using Lemma 3.3 which showed each of the two splitte for a variable structure requires $7f + 6$ background rectangles, the tot over all the splitters is $7l + 12v$, for a total of $7l + 17v$ backgroun rectangles over all variable structures. The joint requires $12l + 2$ and eac line switch requires $12l - 3$. We have c in the connection to the clause and there are $5c$ in all the clause checkers. The total background cove then requires $6c + 19l + 17v + 2 + s(12l - 3)$.

Now we consider the beams. Suppose I is satisfiable. Consider som satisfying truth assignment. Then for each variable u in U, we set th upper beam in the inverter in the corresponding variable structure on if is assigned the value true. Otherwise we set the lower beam in th corresponding inverter on. This requires v beam rectangles.

Now, consider a splitter with fanout f in which the input beam has bee set on. By Lemma 3.3 we are allowed f beams, in addition to the inp one, which must cover each of the beam locks and there must be or beam per beam machine. Thus, the output beams must all be off. On th other hand, when the input is off (which is the case for one of the splitte in each variable), we may use one beam to cover the beam locks and th bottom right beam machine, while using the remaining f rectangles turn all output beams on. Since the total number of output beams is l, v have $2v + l$ beam rectangles in the variables.

For the joint and the line switches, we see that if an input beam is then we may turn on the output beam with one rectangle per beam li per device. Otherwise, we must turn on the joining rectangle to obtain minimal cover. Either way takes l rectangles per device, for a total $sl + l$ rectangles, assuming s line switches and one joint. The total bea rectangles used is then $2v + sl + 2l$, and the total rectangles in the cov is $6c + 21l + 2 + s(13l - 3) + 19v$ as claimed. Since by our mapping t beam entering a clause checker is on when the corresponding literal true, there must be one beam on in every clause checker and by Lemr 3.5 every clause is therefore covered.

To see that a covering of size k implies a satisfying truth assignment, note that for such a covering each clause checker must have at least o

beam on. But working backwards now, this implies the corresponding output beams from some of the variable structures must be on. But from the above argument the splitter involved can be minimally covered with any output beam on only if the corresponding beam locks are covered by the one internal beam, and the input beam is off. But this can be used to assign a truth value to the corresponding variable in I, namely, the corresponding literal is true. The coupling implies every clause must have one true variable under this assignment.

Adding together the construction times above, and noting that $l = 3c$ and $l \geq v$ and $s < l$, we see that the polygon can be constructed in $O(l^2)$ time. □

3.2. *Interior and Boundary Covering in General Position*

In this section we show that covering either the interior or boundary of an orthogonal polygon with rectangles is NP-complete, even if the polygon is required to be in general position. In the previous section we showed that interior covering is NP-complete, but we frequently used alignments to simplify our construction. Thus, the results in this part include the results of the previous section as a special case. This construction is much more complicated, however, so we only describe it in outline, indicating differences from the preceding construction.

It is interesting to note that the problem is NP-complete even when the polygon is in general position. One might conjecture that the many alignments that may arise in the non-general position case should make the problem more difficult. We show that this is not the case.

We first state the definition of the problem.

Rectangle Boundary Cover General Position RBCGP.

Instance: A simple and holeless orthogonal polygon P in general position, and a positive integer k.

Question: Is there a covering of the boundary of P consisting of k or fewer rectangles contained in P?

The construction of the preceding section fails to satisfy this definition in two respects. First, not all of the critical sources are located on the boundary, for example the central source in the beam machine in Fig. 13. It is possible to cover the boundary of the beam machine with fewer rectangles than is required for the interior, and these coverings make the beam machines fail to operate correctly. Second, it often makes use of interior alignments to make the device work properly. We now show that RBCGP is NP-complete.

The construction is similar to the preceding one, except that the devices are more complicated. In Fig. 22 we illustrate the features of the beam

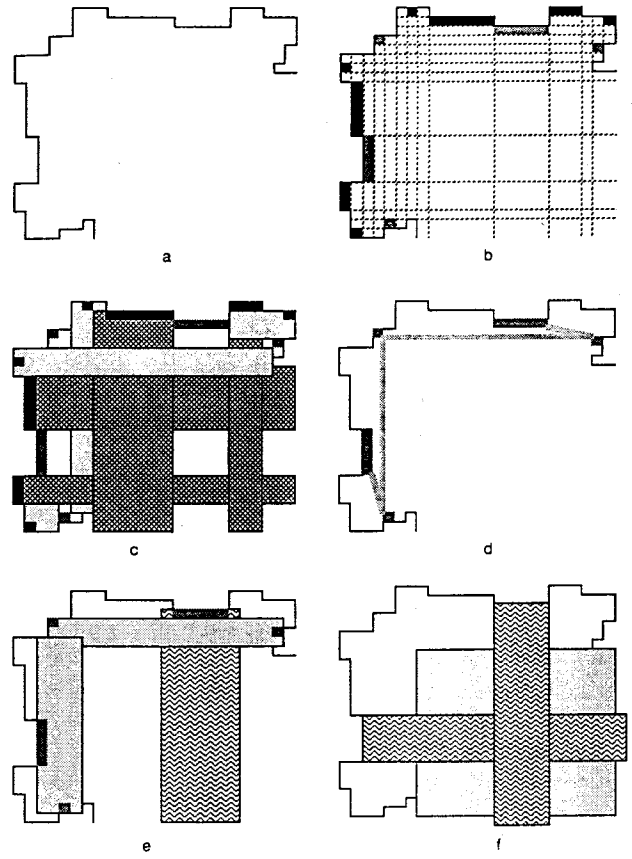


FIG. 22. General position beam machine (see text).

machine. Figure 22a shows the outline of the beam machine, while Fig. 22b shows the cells of the beam machine together with the sources we require. Note that usually the larger sources will be refined by extensions of edges from other devices coupled with the beam machine. The black sources are not visible to any other source in the beam machine. Although portions of the larger black sources will be visible to sources in other devices, the refined portion will not be able to see any other source.

Special note should be made that all of these sources, and other sources we will identify, are located on the boundary of the rectangle.[5] Thus, the

[5]Not all of these sources are sources in the sense of the source graphs in [10], but they are projections onto the boundary of such sources. In fact, they are intersections of portions of the boundary with a "natural source."

difficulty of this problem will depend upon the complexity of covering the boundary. The minimal covering will also cover the interior of the polygon. This constraint makes the construction more complicated.

In Fig. 22c we show a covering for the black sources. Note that the light gray rectangles are the unique maximal rectangles covering their sources. The hashed rectangles can be extended through the beam mouth to cover parts external to the beam machine. We refer to these rectangles as the *beam rails*. The *beams* are the regions between the beam rails.

The remaining gray sources must also be covered by rectangles. The visibilities between the gray sources are illustrated in Fig. 22d. These sources can be covered in pairs just as in the previous beam machines, with one remaining to be covered by a rectangle that can be extended between (and overlapping) the beam rails in exactly one of the orthogonal directions. In Fig. 22e one such covering of the gray sources is shown, with the beam extended downwards. Figure 22f shows the positions of the two beams by rectangles shaded with a wavy pattern, together with the mouth region of the beam machine shaded in gray.

We should point out to the careful reader that we occasionally cheat in drawing our beam machines and other devices. Careful measurement would show that that some of the edges appear to be aligned, and thus there may be more than two vertices in a horizontal or vertical line. We are using a grid in the graphics package to make sure the edges of various pieces meet correctly at the end points. However, this restricts the accuracy with which we can place edges. All the resulting alignments are accidental, and have no bearing on the proof. The line joining any aligned vertices passes through regions external to the polygon, and so the edges may be perturbed out of alignment without changing the visibility conditions between the identified sources. That is, we have eliminated all internal alignments which could be useful in our construction.

In Fig. 23 we illustrate a technique we use to ensure coverings of the interior using sources located on the boundary in regions joining various devices. Note that each rectangle is the unique maximal rectangle covering its source. We refer to this structure as a *shiplap*, as each rectangle overlaps the one next to it. The beam rails in the beam machines are used to terminate the shiplaps.

We illustrate with a very simple example. Fig. 24 shows how two beam machines can be coupled to transmit a signal. We hasten to point out that in the more complex structures to follow, the beam machines will be much further apart horizontally to allow other beams to pass between them. Fig. 24a shows the device, together with the sources defining the beams. Notice the refinement of the sources defining the shared horizontal beam. In Fig. 24b we show how the beam rails are linked. Note how the sources are refined for those beam rails which are restricted internally to the device.

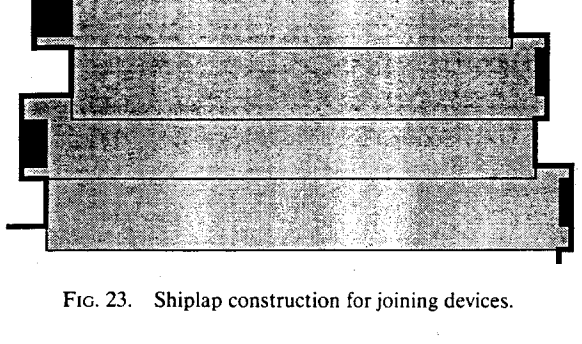


FIG. 23. Shiplap construction for joining devices.

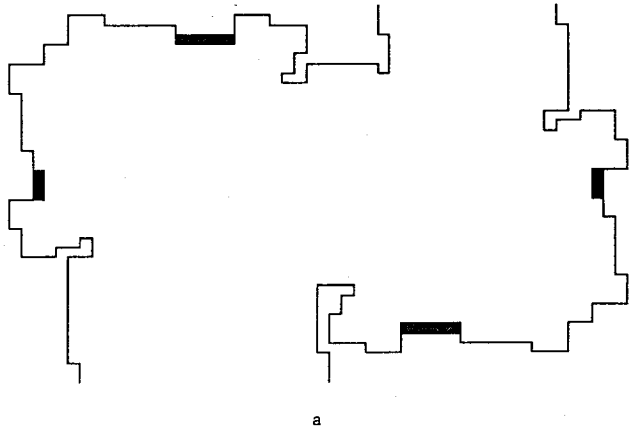
Similar refinements would occur for the rails leaving the device, once th remaining devices are in place.

In Fig. 24c the remaining background cover is shown by gray rectangle while the beams are shown using a wavy pattern. Notice that the gra rectangles are the unique maximal rectangles covering their sources. On further rectangle, shaded by a striped pattern, is required, and may be beam rail if the next device is also a beam machine. This covers an interic region that would otherwise be uncovered if the joining beam is off. Th source generating this rectangle is not shown, but would be located on th boundary of the next device coupled to this somewhere below the diagran The background rectangles, including the beam rails, together with eithe the interior joining beam or the input and output beams are sufficient an necessary to cover the boundary of the device as well as the interior.

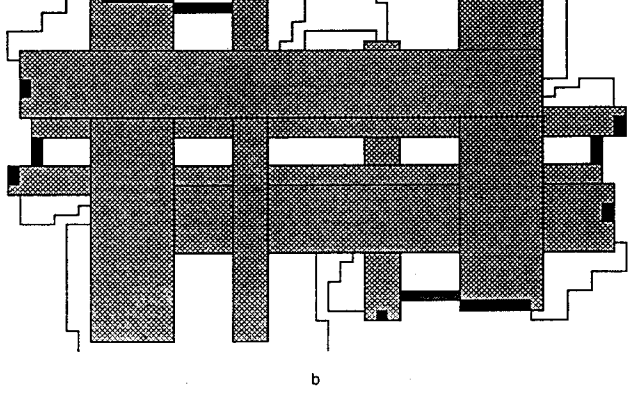
The next device of interest is the beam splitter. Since all sources mus be on the boundary and we are not allowed alignments, we can cover a most one source on each side of any covering rectangle. This places a upper bound on the degree of splitting a single device can accomplish. I our device, shown in Fig. 25 with background cover in Fig. 26, we can spl a beam into two output beams. However, this does not place an uppe bound on the fanout, since we can always use more splitters to split th output beams, as described in the following paragraphs.

The operation of this beam splitter is similar to the splitter in th previous section.

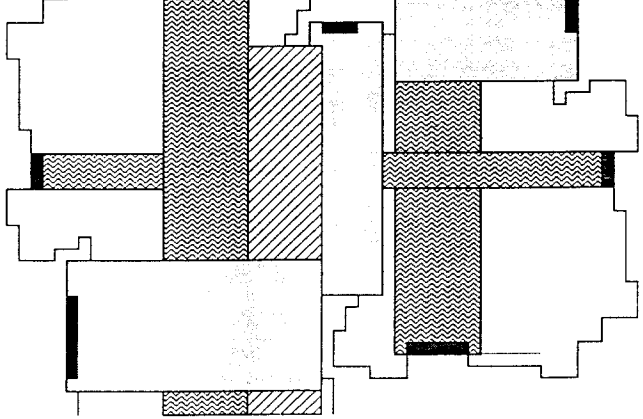
In Fig. 27 we show the basic variable structure with fanout of two o each side. To increase the fanout, we build a *splitting joint* somewhat lik the joint described in the previous section, but for general position polygons. On the right side of the splitting joint we will substitute bear splitters for beam machines on any beam line we wish to split. Additiona splitting joints may be used until the desired fanout is achieved for th output from each side of the variable structure. Each splitter increases th



a



b



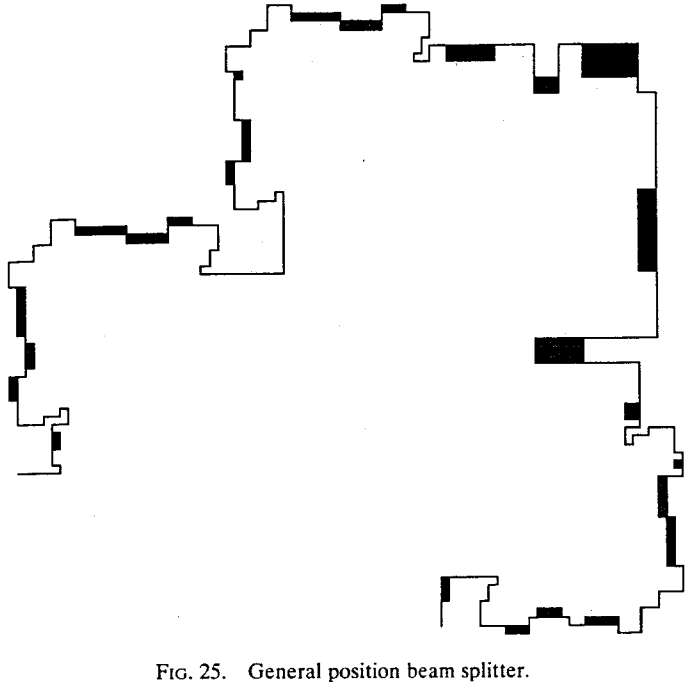


FIG. 25. General position beam splitter.

fanout by one, so it takes $f - 1$ splitters to achieve a fanout of f, and thus the construction is polynomial. (The number of splitting joints is logarithmic in f.) We note that the on/off vertical/horizontal parity of the beam splitters is the same as that of the beam machines, so the output of the splitting joint is correct.

At the left of the variable structure in Fig. 27 is an inverter, similar to the one in the previous section. In Fig. 28 we show the background rectangles required to cover the variable structure and the inverter, except the switch rectangle. Note that there is only a single switch source at the top of the inverter, which is visible from either of the beam sources. Thus as for the inverter of the previous section, one incident beam must be on to minimally cover this inverter.

A 3-clause checker can be constructed by adding one more switch source, and one more beam source to an inverter, similar to the construction in the previous section.

The general position line switch is illustrated in Figs. 29 and Fig. 30. Here the second of four input beam lines is switched to the third position. Some of the internal edge extensions in Fig. 29 are shown to aid in establishing the relationship between sources. An additional beam ma-

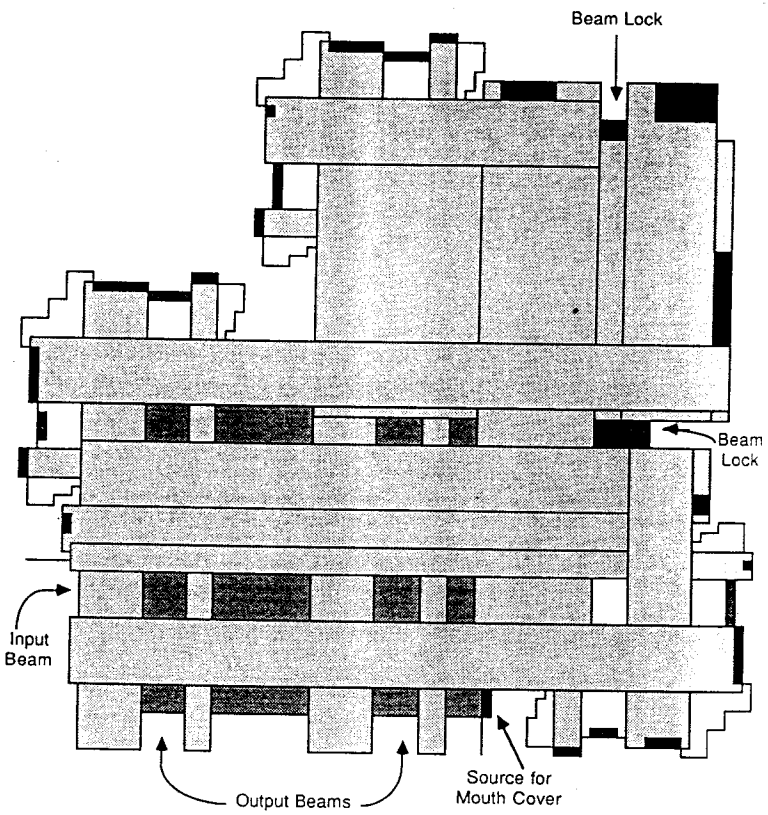


FIG. 26. Background for general position beam splitter.

chine is shown above the device to illustrate how the input switch beam is aligned. All of the horizontal shiplap structure is shown, and most of the vertical. On the input side, the striped rectangles show where the input beam rails extend, and the three with stripes of opposite slant are required; these three will be beam rails from beam machines above or rectangles from additional shiplap dents between beam machines above the line switch. Similarly, additional dents for shiplap structure can be inserted between the beams on the output side of the line switch to complete covers for the device(s) below the line switch as required by simply extending the horizontal distance between the beam machines and inserting notches.

The line switch works much as the one in the previous section. At the top are three notches creating sources which are visible to four sources on the left side. Two of the left side sources (labeled as background interior

FIG. 27. General position variable structure.

sources in Fig 30) are not visible to any other sources, while one is visible to an input beam source and one is visible to a joining beam source. Thus at least one of those beams must be on if the line switch is to be covered minimally. Additional background sources are located at the left side o the output end of the device. These background sources are created or the boundary so that a boundary cover will cover the interior. They are necessary since the regions where two beams cross would not otherwise be covered if both beams are off.

Counting the number of rectangles in a minimum cover under the assumption that the instance of 3SAT is satisfiable is complicated by the fanout construction. Basically, for each variable we must determine how many beam machines and how many splitters are required, as well as account for the background for connecting these different devices. However, given any instance and its corresponding construction, it is straightforward if tedious to determine this number. We leave this as well as the remaining construction details as an exercise for the masochistic reader.

This construction shows that:

THEOREM 3.2. *RBCGP is NP-complete.*

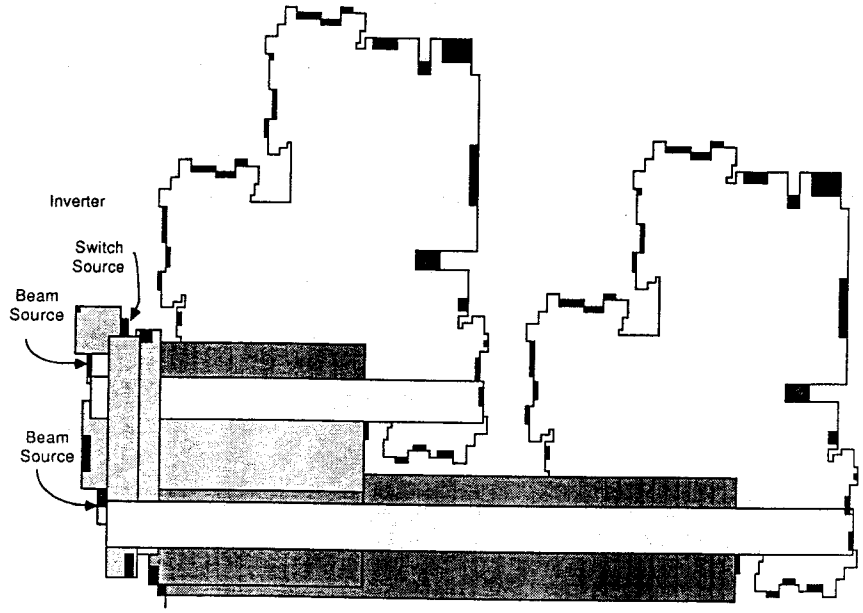


FIG. 28. Background for general position variable structure.

4. CONCLUSION

We have shown that minimum convex covering of the interior or the boundary of arbitrary polygons are NP-hard problems, and covering the vertices is NP-complete, even when the given polygon is required to contain no holes and be in general position. We have also shown that minimal rectangle covering of the interior or boundary of an orthogonal polygon is NP-complete, even when the polygon is required to contain no holes and be in general position. We are unable to show that general convex covering is in NP. In fact, it is not known whether this problem is even in P-space, although O'Rourke [24] has shown that it is decidable using Tarski's method. Little is known about approximation algorithms for the minimal convex covering problem. Aggarwal *et al.* consider the problem [2], but restrict the covers to pieces whose edges are segments containing two vertices of the polygon. They show that under this restriction there can be exponentially many maximal convex pieces. They give an approximation algorithm for star covers under this restriction.

An orthogonal polygon is *orthogonally convex* (OC) if every orthogonal line intersects it in at most one connected segment. It is *horizontally* (equivalently, *vertically*) *convex* if every horizontal (vertical) line intersects

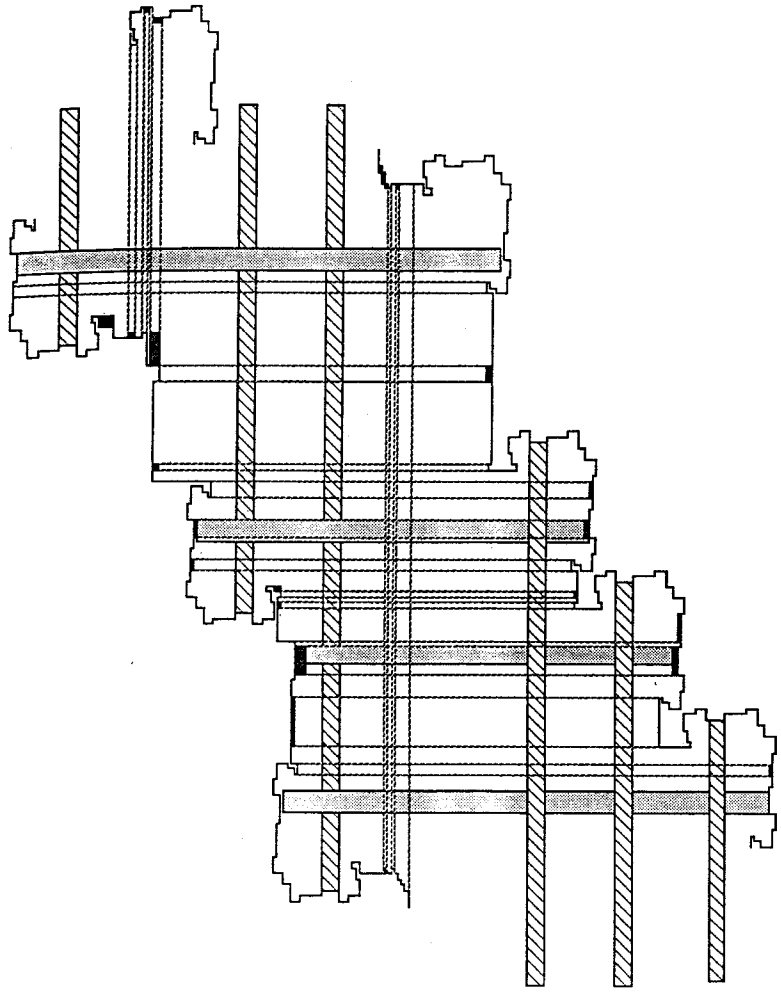


FIG. 29. General position line switch.

it at most once. Thus, every orthogonally convex polygon is horizontally convex. A problem closely related to rectangle covering that has received attention in the literature is the problem of covering orthogonal polygons with orthogonally convex polygons [16, 8, 26]. We have been unable to construct a beam machine with the necessary properties for this case. In the rectangle construction, the beams are symmetric from end to end, and they are focused so that they can cover exactly one small region at each end. These joint properties do not seem to be possible where orthogonally convex covers are concerned.

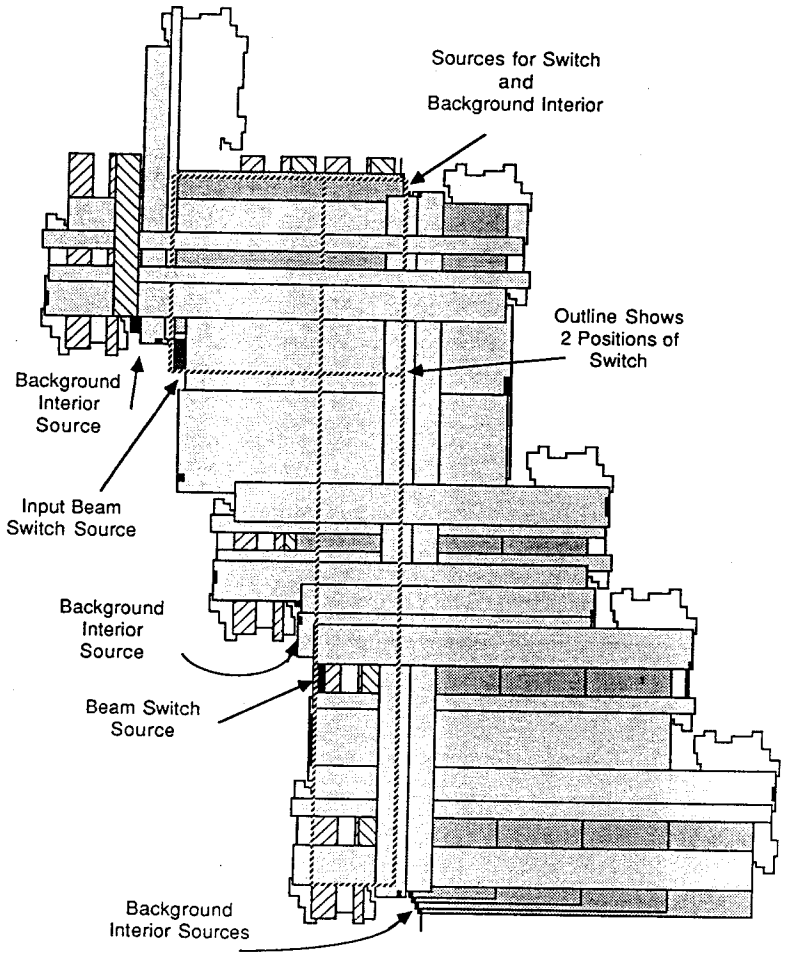


FIG. 30. Background for general position line switch.

In arbitrary polygons, covering by star-shaped polygons is equivalent to placing guards to see all points in the polygon. This is true since a star is made up of the union of convex polygons which have some point in common. The set of such points is the *kernel* of the star. The minimum guard problem can be solved by placing guards in the kernels of the minimum star covering, and vice versa.

An *orthogonal star* (OS) is an orthogonal polygon formed from the union of rectangles which have a common intersection. Equivalently, in an orthogonal star S, there exists a point k such that for every point i in S, there is a rectangle contained in S containing both k and i. Keil [16]

showed that covering horizontally convex orthogonal polygons with orthog onal stars can be done in $O(n^2)$ time. Franzblau and Kleitman [12] gave a $O(n^2)$ algorithm for covering vertically convex (equivalent to horizontall convex) orthogonal polygons with rectangles. It is as yet unknown whethe there is a polynomial time algorithm for covering general holeless orthogo nal polygons with OS polygons, nor has it been shown to be NP-hard.

Finally, we can define an *orthogonally convex star* (OCS) as the union c orthogonally convex polygons which have a common intersection. Equiva lently, an OCS contains a kernel k such that for every point p in the OCS there is an OC containing k and p. Note that this definition allows a OCS to have holes, but if the polygon being covered has no holes, then w can eliminate the holes from the covering OCS polygons. Motwan *et al.* [22] have shown that orthogonal polygons without holes can b optimally covered by OCS polygons in polynomial time. Fast algorithm for both OC and OCS coverings for special cases of orthogonal polygon exist [16, 8, 10].

We note that every rectangle (R) is trivially an OS, and that every OC i an OCS. Every OS is also an OC and so we have the class heirarch expressed by

$$R \subset OS \subset OC \subset OCS.$$

We also note that for many instances where an algorithm for covering by convex class exists, then there exists an algorithm for covering by the corresponding stars and similar complementary results for NP-hardnes results, with the two most notable exceptions being the most general case of OS and OC coverings. Coupled with the general interest in determinin the boundary between NP and P (if it exists) this renders these tw problems of considerable interest.

REFERENCES

1. A. AGGARWAL, "The Art Gallery Theorem: Its Variations, Applications and Algorithmic Aspects," Ph.D. thesis, Dept. Elect. Engin. and Comp. Sci., Johns Hopkins Univ., 1984.
2. A. AGGARWAL, S. K. GHOSH, AND R. K. SHYAMASUNDAR, Computational complexity of restricted polygon decompositions, *in* "Computational Morphology" G. T. Toussaint, Ed., "Machine Intelligence and Pattern Recognition," Vol. 6, Chap. 1, p. 1–11, North-Holland, Amsterdam, 1988.
3. S. CHAIKEN, D. J. KLEITMAN, M. SAKS, AND J. SHEARER, Covering regions by rectangles, *SIAM J. Algebraic Discrete Methods* **2**, No. 4 (Dec. 1981), 394–410.
4. B. CHAZELLE, "Computational Geometry and Convexity," Ph.D. thesis, Dept. Comput. Sci., Carnegie–Mellon Univ., July 1980.
5. B. CHAZELLE AND D. DOBKIN, Decomposing a polygon into its convex parts, *in* "Proc. 11th Ann. ACM Symp. Theory of Comp., 1979," pp. 38–48.

6. H. E. CONN AND J. O'ROURKE, Some restricted rectangle covering problems, *in* "Proceeding of the 25th Allerton Conference on Communication, Control and Computing, 1987," pp. 898–907.
7. S. A. COOK, Linear time simulation of deterministic two-way pushdown automata, *in* "Proc. IFIP Cong. 71, Foundations of Information Processing, 1971."
8. J. CULBERSON AND R. A. RECKHOW, Orthogonally convex coverings of orthogonal polygons without holes, *J. Comput. Systems Sci.*, **39** (Oct. 1989), 166–204.
9. J. C. CULBERSON AND R. A. RECKHOW, Covering polygons is hard preliminary abstract, *in* "29th Ann. Symp. Foundations of Comp. Sci., White Plains, NY, Oct. 1988," Vol. 21, pp. 601–611.
10. J. C. CULBERSON AND R. A. RECKHOW, "A Unified Approach to Orthogonal Polygon Covering Problems via Dent Diagrams," Tech. Rep. TR 89-6, Dept. of Comp. Sci., Univ. of Alberta, Feb. 1989.
11. D. S. FRANZBLAU, Performance guarantees on a sweep-line heuristic for covering rectilinear polygons with rectangles, *SIAM J. Discrete Math.* **2**, No. 3 (Aug. 1989), 307–321.
12. D. S. FRANZBLAU AND D. J. KLEITMAN, An algorithm for constructing regions with rectangles: Independence and minimum generating sets of collections of intervals, *in* "Proc. 16th Ann. ACM Symp. Theory of Comp., Apr. 1984," pp. 167–174.
13. M. R. GAREY AND D. S. JOHNSON, "Computers and Intractability: A Guide to the Theory of NP-Completeness," Freeman, San Francisco, 1979.
14. J. M. KEIL, "Decomposing Polygons into Simpler Components," Ph.D. thesis, Dept. of Comp. Sci., Univ. of Toronto, 1983.
15. J. M. KEIL, Decomposing a polygon into simpler components, *SIAM J. Comput.* **14** No. 4 (1985), 799–817.
16. J. M. KEIL, Minimally covering a horizontally convex orthogonally polygon, *in* "Proc. 2nd Ann. Symp. Comp. Geom., June 1986," pp. 43–51.
17. J. M. KEIL AND J. R. SACK, Minimum decompositions of polygonal objects, *in* "Computational Geometry" (G. T. Toussaint, Ed.), North-Holland, Amsterdam, 1985.
18. A. LINGAS, The power of non-rectilinear holes, *in* "9th Int. Colloq. Autom. Lang. Prog.," 1982.
19. W. LIPSKI, E. LODI, F. LUCCIO, C. MUGNAI, AND L. PAGLI, On two dimensional data organization, ii, *Fund. Inform.* **2** (1979), 245–260.
20. A. LUBIW, "Orderings and Some Combinatorial Optimization Problems with Geometric Applications," Ph.D. thesis, Univ. of Toronto, 1985.
21. W. J. MASEK, Some NP-complete set covering problems, unpublished manuscript, Aug. 1979.
22. R. MOTWANI, A. RAGHUNATHAN, AND H. SARAN, Covering orthogonal polygons with star polygons: The perfect graph approach, *in* "Proc. 4th Ann. Symp. Comp. Geom., June 1988," pp. 211–223.
23. T. OHTSUKI, M. SATO, M. TACHIBANA, AND S. TORII, Minimum partitioning of rectilinear regions, *Trans. Inform. Proc. Soc. Japan* (1983).
24. J. O'ROURKE, The complexity of computing minimum convex covers for polygons, *in* "Proceedings of the 20th Annual Allerton Conference on Communication, Control and Computing, 1982," pp. 75–84.
25. J. O'ROURKE AND K. J. SUPOWIT, Some NP-hard polygon decomposition problems, *IEEE Trans. Inform. Theory* **IT-29**, No. 2 (1983), 181–190.
26. R. A. RECKHOW AND J. CULBERSON, Covering a simple orthogonal polygon with a minimum number of orthogonally convex polygons. *in* "Proc. 3rd Ann. Symp. Comp. Geom., June 1987," pp. 268–277.
27. T. SHERMER, Convex cover is NP-hard, private communication, Oct. 1988.

JOURNAL OF ALGORITHMS **17**, 45–70 (1994)

Three Stacks

MICHAEL L. FREDMAN*

Rutgers University, Dept. of Computer Science, New Brunswick, New Jersey 08903

AND

DEBORAH L. GOLDSMITH*

MITRE Corporation, San Diego, California 92152

Received December 1988; revised October 3, 1991

Maintaining two stacks in memory is easily accomplished by placing them at opposite ends of memory and having them grow towards each other. The entire memory is readily available for stack items with this approach. Maintaining three stacks, however, presents more difficulty. Traditionally, the allocation of storage for three stacks has been accomplished by using pointers to store the stacks as linked lists, or by relocating the stacks within memory when collisions take place. The former approach requires additional space to store the pointers, and the latter approach requires additional time. We explore the extent to which either some amount of additional space or time is required to maintain three stacks. We provide a formal setting for this topic and establish upper and lower complexity bounds on various aspects.

1. INTRODUCTION

Maintaining two stacks in memory is easily accomplished by placing them at opposite ends of memory and having them grow toward each other. The entire memory is readily available for stack items with this approach. Maintaining three stacks, however, presents more difficulty. Traditionally, the allocation of storage for three stacks has been accomplished by using pointers to store the stacks as linked lists, or by relocating the stacks within memory when collisions take place. The former approach

*Supported in part by the National Science Foundation under Grant DCR-8504245. This work was completed under the auspices of the University of California at San Diego and Bell Communications Research, Morristown, NJ 07960.

0196-6774/94 $6.00