9

MINIMAL GUARD COVERAGE

9.1. INTRODUCTION

In Chapter 1 it was shown that Fisk's proof of the art gallery theorem can be converted into an algorithm that covers a polygon of n vertices with $\lfloor n/3 \rfloor$ guards in $O(n \log n)$ time. Although $\lfloor n/3 \rfloor$ is necessary in some cases, often this is far more guards than are needed to cover a particular polygon. For example, convex polygons only require one guard, but Avis and Toussaint's algorithm would still place $\lfloor n/3 \rfloor$ guards. It is natural, then, to seek a placement of a *minimal* number of guards that cover a given polygon. We will show, however, that this problem is fundamentally intractable: it is NP-complete.

Finding the minimal number of guards to cover a polygon is a specific instance of a general class of problems on which there is now a considerable literature: polygon decomposition problems. Guards determine star polygons, so minimal guard coverage corresponds to finding a minimal star cover of a polygon. Polygon decomposition problems can be classified along four "dimensions": decomposition type, the shape of the pieces, restrictions on the boundaries of the pieces, and the shape of the polygon being decomposed. We will discuss these dimensions briefly before focusing on our particular case.

(1) Decomposition Type

The term "decomposition" is usually understood to encompass two major types: *partition,* in which the pieces are not allowed to overlap, and *cover,* where overlap is permitted. In both cases, each piece of the decomposition must be a subset of the original polygon, and the union of the pieces must be precisely the polygon. Covers can be significantly more efficient than partitions: Fig. 9.1 shows an example due to Ntafos (1986) that has a star cover of size 2 but the minimum star partition is of size $O(n)$. Since guard lines-of-sight can pass through one another freely, the minimum guard placement problem is a minimal cover problem.

A third type of decomposition, *sum-difference* decomposition, that

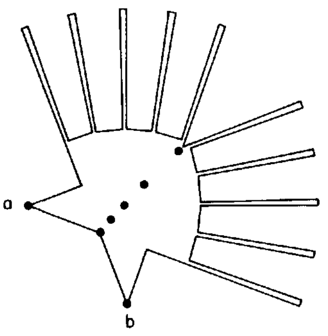


Fig. 9.1. A polygon that may be covered with 2 stars with kernels at *a* and *b*, but that requires 5 stars in a partition, with kernels at the indicated points.

permits both union and difference of pieces, has not been studied yet to the same extent as covers and partitions (Woo 1982).

(2) Shape of Component Pieces

The major shape types that have been studied are convex (Pavlidis 1968, 1977; Chazelle 1980; O'Rourke 1982a, 1982b), star (Maruyama 1972; Avis and Toussaint 1981a), spiral (Feng and Pavlidis 1975; Pavlidis and Feng 1977), monotone (Lee and Preparata 1977), and trapezoidal (Asano, Asano, and Imai 1986). For orthogonal polygons, both rectangle (Pagli *et al.* 1979; Chaiken *et al.* 1981; Franzblau and Kleitman 1984) and square (Albertson and O'Keefe 1981) pieces have been examined; L-shaped partitions were discussed in Sections 2.5 and 2.6. We will only discuss star pieces.

(3) Restrictions on the Boundaries of the Pieces

In addition to the shape restrictions on the pieces, two further restrictions that cut across shape types are important. A *Steiner point* is any point in a polygon that is *not* a vertex.[1] Decompositions are then classified as *with Steiner points* permitted, or *without Steiner points*. The latter are only permitted to use vertex-to-vertex diagonals to compose the boundaries of the pieces. In general, decompositions with Steiner points are more efficient, and harder to find. Figure 9.2 shows a polygon that can be covered with two convex pieces if Steiner points are permitted, but which requires three pieces without Steiner points. A minimum convex partition without Steiner points has no more than four times the number of pieces in a minimal partition with Steiner points, as is established, for example, by the Mehlhorn and Hertel triangulation argument mentioned in Section 1.3.2. (It does not seem to be known if this worst-case bound can be achieved.) The situation for star pieces is less clear. I know of no results comparing the

1. Steiner used such points to define what is now known as a *Steiner tree*, a minimal spanning tree employing points in addition to those being spanned.

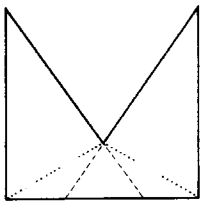


Fig. 9.2. A polygon that may be covered with 2 convex pieces if Steiner points are permitted (dashed), but requires 3 if not permitted (dotted).

efficiency of partitions or covers of star pieces with and without Steiner points.

Other restrictions on the boundaries of the component pieces of decompositions have been considered. The most interesting is to restrict all edges to be subsets of extensions of the polygon edges through the interior of the polygon. This was studied by Pavlidis (1968, 1977) and myself (O'Rourke 1982a,b) for convex partitions, and by Aggarwal *et al.* (1985) for star covers. Figure 9.3 shows an example of the latter authors that requires only two star pieces if diagonals are permitted, but needs three if only edge extensions are used.

(4) Polygon Restrictions

The two most important classes here are polygons with and without holes. In at least one case, convex partitions with Steiner points, the polygon problem can be solved in polynomial time (Section 1.4.2), but permitting holes changes the complexity to NP-hard (Masek 1979) (O'Rourke and Supowit 1983). We will discuss both cases for star covers in this chapter.

Of course, many other polygon restrictions can be considered. We will briefly discuss the restriction to orthogonal polygons.

Our focus in this chapter is minimal star covers with Steiner points. We show in the next section that this problem is NP-hard for polygons with holes, and in Section 9.3 it is shown that the problem remains NP-hard for polygons without holes. Clearly the latter result supersedes the former, but the proofs are quite different, and we present them in the order in which they were discovered. Next we look at two restrictions that permit polynomial-time solutions: guards in "grids," special orthogonal polygons (Section 9.4), and minimal star partitions without Steiner points (Section

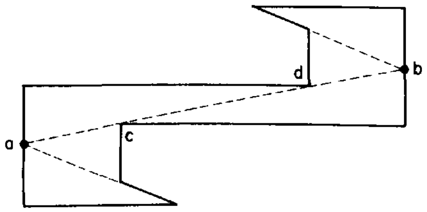


Fig. 9.3. A polygon that can be covered with 2 stars if diagonal *cd* is used, but that requires 3 if only edge extensions are used.

9.5). The former problem can be solved with a graph matching algorithm, and the latter with dynamic programming. Several related algorithms and problems are discussed in the final section.

9.2. NP-HARD FOR POLYGONS WITH HOLES[2]

Although it would take us too far afield to explain the extensive theory of NP-completeness, we will sketch enough of the definitions so that the next two sections can be followed by the uninitiated.

The theory hinges on a division of problems into two complexity classes: *P* and *NP*. Problems in *P* can be solved in deterministic *P*olynomial time, and problems in *NP* can be solved in *N*on-deterministic *P*olynomial time. The addition of the power of non-determinism seems to widen the class of problems considerably, but although $P \subseteq NP$ follows from the definitions, no one has been able to prove that $P \neq NP$. This is, in fact, the major open problem in computer science today. Despite this uncertainty, the hardest problems in *NP*, the *NP-complete* problems, seem fundamentally intractable: although over 1000 NP-complete problems have been identified, no polynomial algorithm is known for any of them.

The "hardness" partial order is established by the notion of *polynomial reducibility*. If problem A can be transformed in polynomial time to an instance of problem B such that the solution to B yields a solution to A, then A is said to be polynomial reducible, or just reducible, to B. If A is reducible to B, then B is at least as hard as A, for if B can be solved in polynomial time, so can A. A problem $B \in NP$ to which all problems $A \in NP$ can be reduced is called *NP*-complete: it is at least as hard as any problem in *NP*. If an NP-complete problem B is reducible to a problem C, then C is called *NP*-hard: it is at least as hard as an *NP*-complete problem. The distinction between NP-complete and NP-hard problems is that the former are members of NP but the latter need not be.

As the classes *P* and *NP* are defined for *decision* problems—that is, those that output only "yes" or "no"—we will phrase the minimum guard cover problems as decision questions also. Since it is important to identify clearly the size of the input, we will specify the problems in Garey and Johnson's instance-question format (Garey and Johnson 1979). Our proofs of NP-hardness will start from a known NP-complete problem, Boolean 3-Satisfiability. It may be stated as follows.

Boolean 3-Satisfiability (3SAT)

INSTANCE: A set $U = \{u_1, u_2, \ldots, u_n\}$ of Boolean variables and a collection $C = \{c_1, c_2, \ldots, c_m\}$ of clauses over U such that each $c_i \in C$ is a disjunction of precisely three literals.

2. This section is the result of joint work with Ken Supowit; an earlier version appeared in O'Rourke and Supowit (1983), © 1983 IEEE.

QUESTION: Is there a satisfying truth assignment for C, that is, is there a truth assignment to the n variables in U such that the conjunctive normal form $c_1 \cdot c_2 \cdots c_m$ is true?

The minimum guard cover problem is formally stated as follows.

Minimum Star (or Guard) Cover of a Polygon with Holes (StarCH)

INSTANCE: A set of lists of integer-coordinate vertices representing a polygonal region P with holes, and a positive integer bound K.

QUESTION: Is there a cover of P with K or fewer star subsets of P, i.e., do there exist star polygons $S_1, S_2, \ldots, S_k$ with $k \leq K$ such that $S_1 \cup S_2 \cup \cdots \cup S_k = P$?

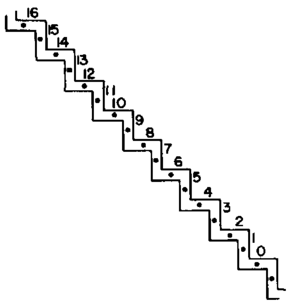
The goal is now to polynomially transform a given instance of 3SAT into a polygonal region that has a star cover of K or fewer pieces iff the 3SAT instance is satisfiable. The proof proceeds along lines similar to proofs of NP-completeness by Fowler *et al.* on the "box-cover" problem (Fowler *et al.* 1981), and by Supowit on point and disk coverage problems (Supowit 1981).

The usual first step in a proof of NP-completeness is to show that the problem is a member of the class of NP problems, that is, solvable via a non-deterministic algorithm in polynomial time (Garey and Johnson 1979). Often this is easy, merely requiring a demonstration that a solution "guessed" by a non-deterministic program can be checked in polynomial time. With the integer-lattice geometric objects used in our constructions, however, it is unclear how to establish this. In the absence of a proof that StarCH is a member of NP, the argument presented below will establish that StarCH is NP-hard rather than NP-complete.

We will now show that 3SAT is polynomially transformable to StarCH. The goal is to accept an instance of 3SAT as input and construct, in polynomial time, a polygonal region that has a cover with a certain number K or fewer star polygons iff the given set of clauses is satisfiable. As with other 3SAT transformations (Garey and Johnson 1979; Fowler *et al.* 1981; Supowit 1981), the construction forces a truth assignment with n "truth-setting components" that simulate the Boolean variables, and ensures satisfaction with m "clause components" that correspond to the disjunctive clauses.

Truth-Setting Components

The truth-setting components are polygonal regions of a repetitive staircase pattern that have just two distinct minimum star covers; see Fig. 9.4. One of the minimum covers is associated with the truth assignment *true* and the other with *false*. There are certain distinguished points associated with each pattern, labeled with integers in the figures. These points are not part of the construction; they are used in the proof that the polygon constructed has the appropriate cover iff the clause is satisfiable. Note that two of the



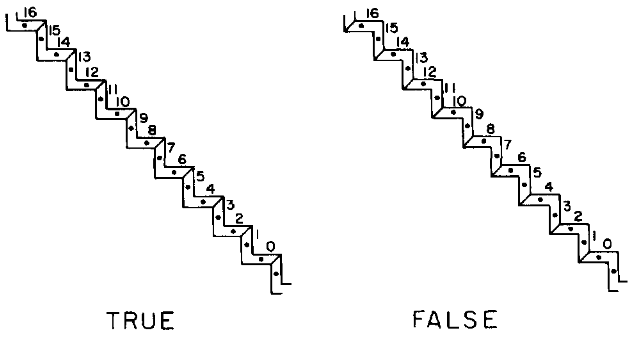


Fig. 9.4. The truth-setting component has two distinct minimum covers corresponding to assignments *true* and *false*.

distinguished points can be covered by one star polygon iff the two points are consecutive in numerical sequence.

The truth-setting patterns are bent (see Lemma 9.2 below) to form closed loops, called *variable loops*. There will be one such loop per Boolean variable u_k in the final construction.

By the above remarks, each minimum star cover for a variable loop contains exactly $r_k/2$ stars, where r_k is the number of distinguished points in the variable loop corresponding to u_k. We call such a cover *true* if it contains distinguished points i and $i+1$ for all even i (taken modulo r_k), and we call it *false* if it contains distinguished points j and $j+1$ for all odd j. Define the bound K used in the definition of the StarCH problem as equal to $\frac{1}{2}\sum_{k=1}^{n} r_k$.

The main properties of variable loops are stated somewhat informally as follows (the proofs, being straightforward, are either omitted or sketched):

LEMMA 9.1. Each minimum star cover is either a *true* cover or a *false* cover.

In constructing the polygonal region P, it will be necessary to cross variable loops over one another, and "bend" them 45°, without these

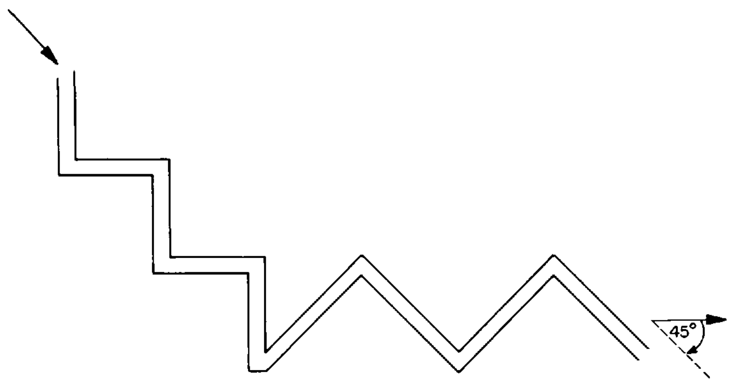


Fig. 9.5. Variable loop 45° bend.

modifications affecting the truth of Lemma 9.1 for any variable loop. The ability to bend a variable loop effectively gives us what is sometimes called an "inverter" in other NP-completeness constructions (Masek 1979).

LEMMA 9.2. The variable loops may bend 45° without affecting their properties.

Proof. See Fig. 9.5. Note that it is not difficult to make all vertices have integer coordinates. □

LEMMA 9.3. Two variable loops may cross over one another without affecting their independent coverage properties. More precisely, if two unconnected variable loops require k_1 and k_2 stars in a minimum cover, then one can cross over the other in such a manner that the resulting (connected) polygonal region requires $k_1 + k_2$ star pieces in a minimum cover, and without altering the type of coverage (*true/false*) within either variable loop.

Proof. The construction is shown in Fig. 9.6. It would not be difficult to show that it preserves the desired properties were it not for the possibility that distinguished points i and j may be covered by a single star-shaped polygon Q (shown in shaded in the figure). We can, however, arrange the crossovers to ensure that at each crossover in the complete construction, i and j are both odd. We will argue that this arrangement ensures that each covering that contains no more than K pieces (if there are any) is a *true/false* covering.

Define a *cross star* to be a star polygon containing at least one distinguished point of each of two variable loops—for example, the polygon Q in Fig. 9.6. Since the points i and j are both odd at every crossover, each cross star contains exactly two distinguished points, both of which are odd. Since no star-shaped polygon can contain more than one even distinguished point, and since there are K even distinguished points, every cover must contain K stars to cover the even distinguished points. Therefore, if a cover

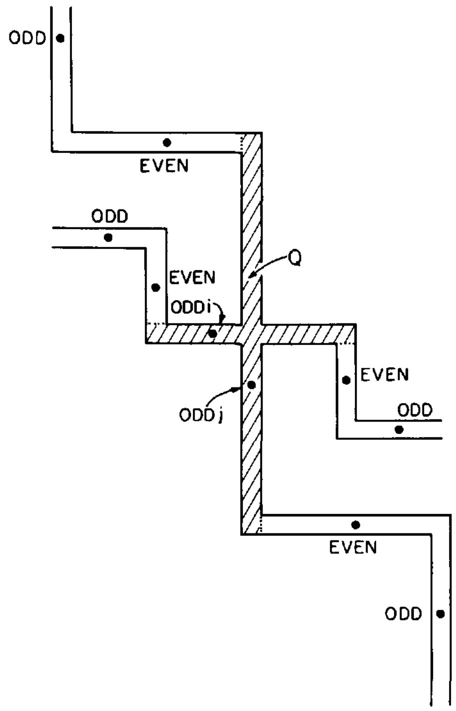


Fig. 9.6. Variable loop crossover. Star Q contains two odd distinguished points.

also includes one or more cross stars, then it must have more than K elements. A similar argument was used in (Supowit 1981).[3] □

Clause Junctions

In a minimum cover of a particular variable loop, small triangular areas near the kernels of the star pieces can be added that could be covered "free" (without increasing the number of pieces) if the coverage is of type *true* (say), but that cannot be covered free if the coverage is of type *false* (see Fig. 9.7). This is the key idea in the formation of the clause junctions.

The heart of a clause junction is an isosceles triangle whose equal sides both slope at 45°. (The shape of this triangle is not critical, but it is easier to keep to integer coordinates if its sides slope at 45°.) Arms of three different variable loops are brought to the junction, one for each side of the triangle. Suppose the clause represented by the junction is $c = u_i + \bar{u}_j + u_k$. Then variable loop j is arranged so that a setting of *true* will permit the triangle to be covered free, and variable loops i and k are placed so that a setting of *false* will result in free coverage. The result is that the triangle at the clause junction will be covered free iff the clause is satisfied by the truth assignment established by the truth-setting components.

3. Use of "planar 3SAT" (Lichtenstein 1982a) apparently obviates the need for this lemma, but it introduces other complications.

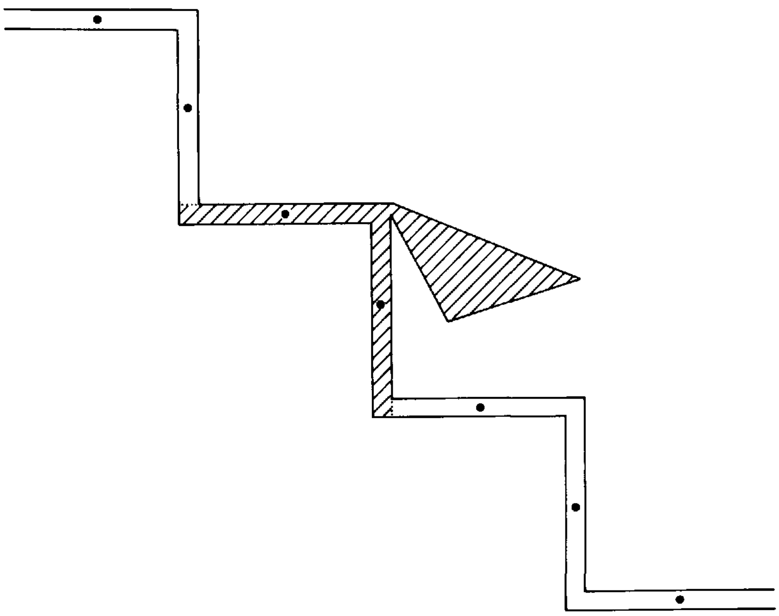


Fig. 9.7. The triangular region is covered free when the truth assignment covers the shaded distinguished points.

The details of clause junction construction are shown in Fig. 9.8. The three important claims concerning the clause junctions are contained in the following proposition.

LEMMA 9.4. The clause junction illustrated in Fig. 9.8 possesses the following properties:

(1) All vertex coordinates are integers.
(2) The central triangle is covered free iff the clause is satisfied.
(3) The junction does not affect the independent coverage properties of the participating variable loops.

Proof. That all vertex coordinates are integers follows from the method of constructing the bends and the use of 45° angles in the clause triangles. Figure 9.7 establishes that the central triangle can be covered free if the clause is satisfied. On the other hand, if the clause is not satisfied, then the construction of the clause junction prevents any piece covering a distinguished point of one of the variable loops to also completely cover the central triangle. Finally, no two distinguished points belonging to different varible loops can be covered by a single piece. Since every piece in a minimum cover must include two distinguished points, there is no interference in their independent coverage properties. □

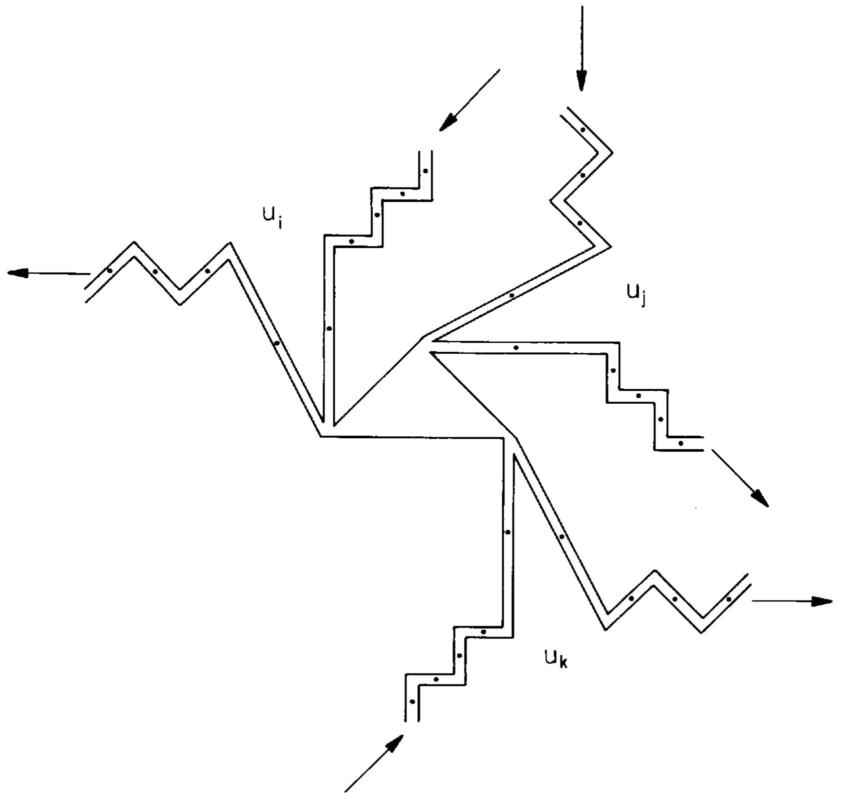


Fig. 9.8. A clause junction. The central triangle is covered free iff the corresponding clause is satisfied.

Complete Construction

The overall structure of the polygonal region constructed for a given instance of 3SAT consists of n variable loops arranged in parallel slanting columns, one for each Boolean variable in U, with m clause junctions placed to the right, one for each clause in C. Arms of the three variable loops corresponding to the three literals that participate in a clause are brought across the other loops to the right, bent in 45° increments until they are oriented properly for the chosen triangle side and according to their complemented/uncomplemented status in the clause, brought to the clause triangle as illustrated previously, and finally returned to their proper slanting columns. A schematic example is shown in Fig. 9.9.

Although the details are complicated, the entire construction can clearly be performed mechanically using the bend, crossover, and clause junction patterns shown previously. The construction requires no more than $O(m)$ bends, $O(mn)$ crossovers, and $O(m)$ clause junctions, so the execution time of the entire procedure is polynomial bounded by $O(mn)$. Note again that

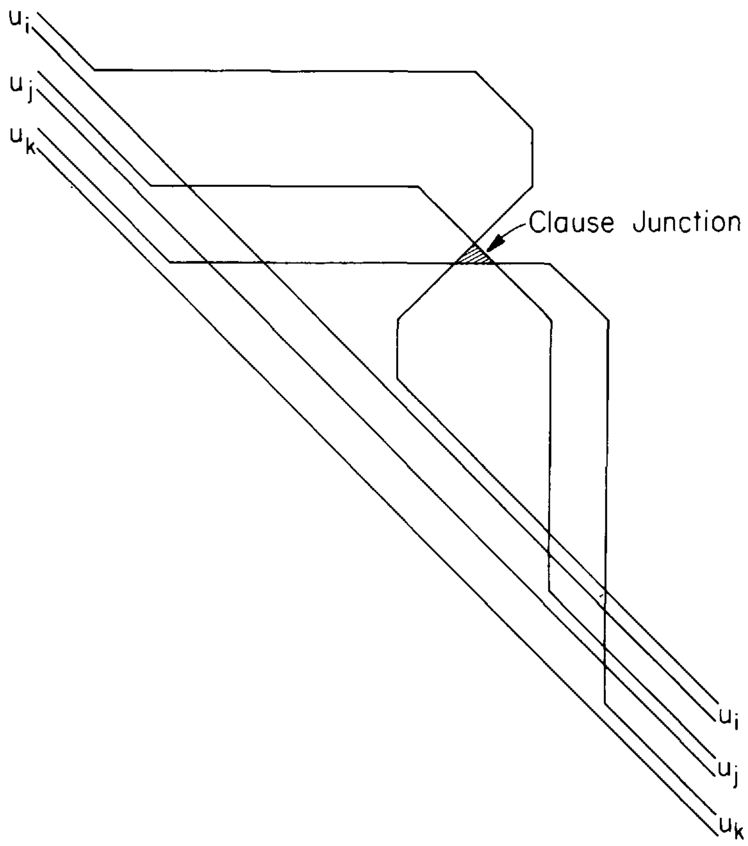


Fig. 9.9. Arrangement of variable loops and a clause junction.

since all of the patterns use integer coordinates, the vertices of the final polygon region will all have integer coordinates. These observations imply the following proposition.

LEMMA 9.5. The construction of the polygonal region requires only polynomial time.

Recall that the bound K was defined to be half the total number of distinguished points. Our argument to this point has shown the following.

LEMMA 9.6. A given set of clauses is satisfiable iff there is a star cover of the constructed polygonal region into K or fewer pieces. (Actually, there can never be fewer than K pieces.)

We may finally state the main result of this section.

THEOREM 9.1 [O'Rourke and Supowit 1982]. The problem StarCH is NP-hard.

Proof. Lemmas 9.5 and 9.6 establish that 3SAT is polynomial transformable to StarCH. Since 3SAT is known to be NP-complete, StarCH is NP-hard. □

COROLLARY. StarCH without Steiner points is NP-hard.

Proof. No Steiner points are needed in any of the constructions. □

9.3. NP-HARD FOR POLYGONS WITHOUT HOLES

The proof in the preceding section constructs a polygonal region with at least n holes, one per variable loop. To prove NP-hardness for polygons with no holes requires, then, a different approach. Recall that for minimum convex covers, the problem is NP-hard with holes but polynomial without, suggesting the same might be true for star covers. But recently Lee and Lin found a clever reduction from 3SAT to a polygon of no holes, proving that finding a minimal star cover (problem StarC) is NP-hard even without holes (Lee and Lin 1986). We present their proof in this section. The proof is simpler if guards are restricted to vertices, that is, if the star kernels always include a vertex; this restriction can be removed later.

The coupling between a variable and its appearance in a clause junction was rather direct in the proof in the previous section: a variable loop is connected to the junction almost as if it were a wire carrying "true" or "false" charge. For a simple polygon construction, the coupling is necessarily more subtle, effected by lines of sight. The truth-setting components of the previous section becomes two distinct components here: a literal pattern, and a variable pattern. The literal patterns appear in clause junctions, and the consistency of the true/false settings of the literals are enforced by the variable pattern.

Literal Pattern

A literal pattern is shown in Fig. 9.10. The distinguished point p shown is visible from vertex t or f. The polygon will be arranged so that no other vertex can see p. As the labels suggest, vertex t will be assigned a guard when the truth value of the literal is *true*, and f when *false*.

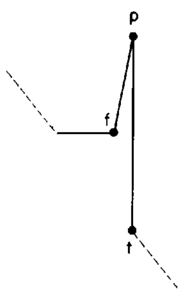


Fig. 9.10. A literal pattern. The distinguished point p is visible only to t and f.

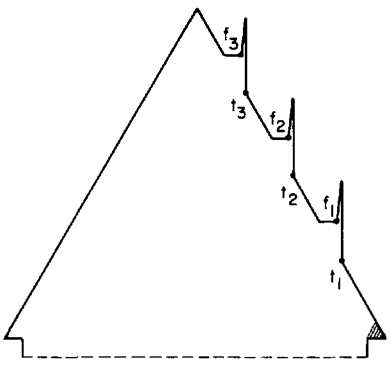


Fig. 9.11. A clause junction. The shaded triangle can be seen only by t_i, $i = 1, 2, 3$.

Clause Junction

A clause junction is shown in Fig. 9.11. Coverage of the three distinguished points in the three literals requires one of $\{t_i, f_i\}$ to be assigned a guard for $i = 1, 2, 3$. At least one of $\{t_1, t_2, t_3\}$ must be assigned a guard in order to cover the shaded triangle. Thus at least one literal must be true for coverage of the junction, which will force each clause to be satisfied.

Variable Pattern

The purpose of a variable pattern is to force all truth assignments of literals of a particular variable to be consistent with one another. This is accomplished with the pattern illustrated in Fig. 9.12. It consists of two "wells," with a vertex at the top of the left well labeled F, and a vertex at the corresponding position on the right well labeled T. In addition each well has s thin spikes, where s is the number of clauses in which the variable u participates, aligned with the F and T vertices and vertices in the clause junctions. One of the two vertices labeled T and F will require a guard in a minimum cover in order to see the distinguished point q illustrated. No other guards will be needed to see the remainder of the variable pattern *if* all the literals for this variable are assigned truth values consistently. This will only become apparent when we examine an example.

Complete Construction

A three variable ($n = 3$), two clause ($m = 2$) example is shown in Fig. 9.13. Here the two clauses are $(u_1 + u_2 + u_3)$ and $(\bar{u}_1 + \bar{u}_2 + u_3)$. Each variable

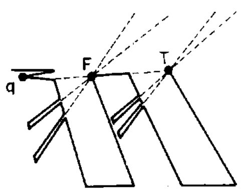


Fig. 9.12. A variable pattern. The distinguished point q can be seen only by T and F.

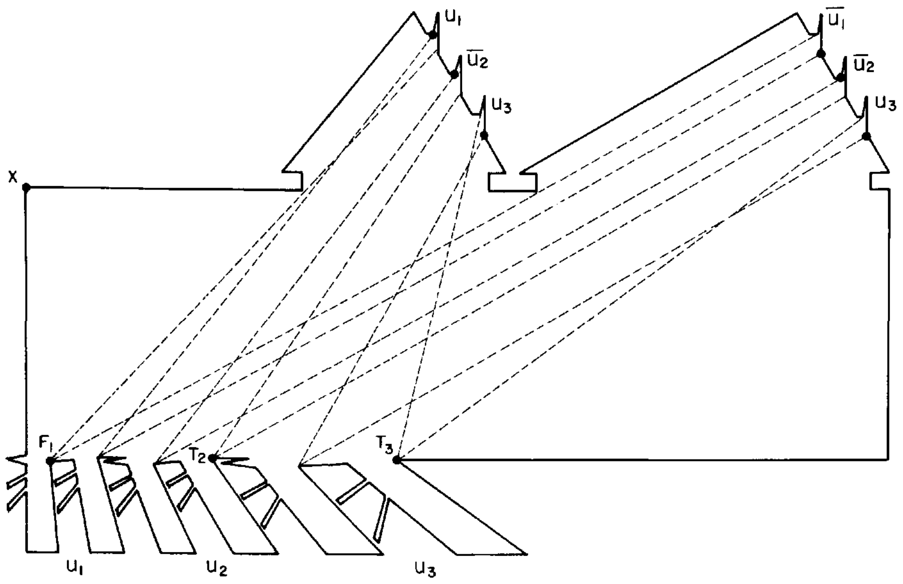


Fig. 9.13. The complete polygon for $(u_1 + \bar{u}_2 + u_3) \cdot (\bar{u}_1 + \bar{u}_2 + u_3)$.

pattern has two spikes, one per well, for each literal in a clause junction that uses that variable. Let u be a variable with distinguished vertices T and F, and let l be a literal in $\{u, \bar{u}\}$ with distinguished vertices t and f. A spike in the left well of the variable pattern for u is collinear with F and t if $l = u$, and with F and f if $l = \bar{u}$. A spike in the right well is collinear with T and f if $l = u$, and with T and t if $l = \bar{u}$. The consequence of these alignments is that a guard placed at F sees down all the spikes of the left well, and a guard placed at T sees down all the spikes of the right well. As mentioned previously, a guard is needed at either T or F to see the distinguished point associated with the variable pattern. Suppose a guard is placed at T, covering all spikes in the right well. Because all spikes in the left well are aligned with F and t_{i_k}, where $t_{i_1}, t_{i_2}, \ldots$ are the t-vertices defined in Fig. 9.11 for all literals using u, and aligned with f_{j_k}, where $f_{j_1}, f_{j_2}, \ldots$ are the f-vertices for all literals $\bar{u}$, all spikes in the left well will be covered if guards are placed at $t_{i_1}, t_{i_2}, \ldots$ and $f_{j_1}, f_{j_2}, \ldots$. This means that if the literals involving u are assigned truth values consistent with $u = \mathit{true}$, then a guard at the F vertex of the u variable pattern is not needed to cover the spikes of the left well, but a guard will be needed at the T vertex. An opposite conclusion is reached for $u = \mathit{false}$. This is the key idea motivating the construction. Thus in Fig. 9.13, u_1 is "set" false by the guard at F_1, and the spikes in the right well of the u_1 variable pattern are covered by a guards at the f-vertex of the u_1 literal in the first clause, and the t-vertex of the $\bar{u}_1$ literal in the second clause.

The total number of distinguished points is $3m + n$: 3 in each of m clause junctions, and 1 in each of n variable patterns.

LEMMA 9.7. A given set of clauses is satisfiable iff the constructed polygon may be covered with $K = 3m + n + 1$ vertex guards.

Proof. If the set of clauses is satisfiable, then there exists a truth assignment to the variables such that each clause is *true*. Placing a guard at the appropraite t or f vertices of the literal patterns in each clause junction necessarily covers the clause junction, because at least one t vertex will be assigned a guard. By the argument presented above, placing a guard at the T vertex of the variable pattern for u if the satisfying truth assignment assigns u *true*, and at F if *false*, covers all the spikes as well as the distinguished point of the pattern for u. Finally, a guard at the vertex x in Fig. 9.13 covers all the wells of all the variable patterns. Thus complete coverage is achieved with K vertex guards.

Suppose there is a cover with K vertex guards. One guard must be at x, otherwise K would not suffice. The remaining $3m + n$ guards are needed to cover the $3m + n$ distinguished points. Each literal pattern must have a guard at either its t or f vertex, and each variable pattern must have a guard at its T or F vertex. Each clause junction will be covered by these guards only if at least one literal pattern has a guard at its t vertex, which implies that each clause will be satisfied. Each variable pattern will be covered by one guard at T or F only if all literals using that variable are assigned consistently, by our previous remarks. But then the guard placement determines a consistent truth assignment that satisfies the given instance of 3SAT. □

THEOREM 9.2 [Lee and Lin 1984]. The minimum vertex guard problem for polygons (StarC) is NP-complete.

Aggarwal extended the argument to obtain the same result for point guards, guards not restricted to vertices (Aggarwal 1984).

COROLLARY [Aggarwal 1984]. The problem of finding a minimum star cover, with Steiner points permitted, for a polygon without holes, is NP-hard.

9.4. GUARDS IN GRIDS

The negative results in the two preceding sections are not the last word on minimal star covers, as there are many interesting special cases that may be tractable. We present two such positive results in this and the following section.

In this section we study a special restricted class of polygons introduced by Ntafos called "grids" (Ntafos 1986); in fact the restriction is so extreme that they are not even polygons. A *grid* P is a connected union of vertical and horizontal line segments. An example is shown in Fig. 9.14. A grid can be thought of as an orthogonal polygon with holes, consisting of very thin corridors. Visibility retains its usual definition: a guard at x can see point y if

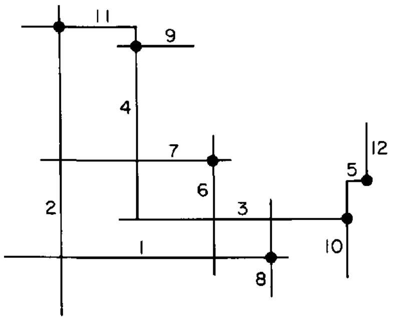


Fig. 9.14. A grid covered by 6 guards (dots).

xy is a subset of *P*. Of course in a grid all lines of sight are either vertical or horizontal, so star polygons are crosses. Although grids are an extreme specialization of polygonal regions with holes, note that a large portion (but not all) of the constructions used to prove Theorem 9.1 could be accomplished with a grid.

Ntafos offered the following simple algorithm for finding a minimal cover by guards in a grid. A guard must be located in each line segment of a grid. Let G be the intersection graph of the grid: each node of G corresponds to a line segment, and two nodes are connected by an arc iff their corresponding segments cross. Figure 9.15 shows the intersection graph for the grid in Fig. 9.14. A *matching* in a graph is a collection of edges M such that every node is incident to at most one edge of M. A *maximum matching* is a matching of maximum cardinality. Notice that a matching in our intersection graph is a collection of intersections that cover pairs of vertical and horizontal segments of the grid. A maximum matching can be found in $O(V^{2.5})$ time for an arbitrary graph of V vertices (Even 1979). The graph G is, however, not arbitrary: it is bipartite, since only intersections between vertical and horizontal segments can occur; two vertical or two horizontal segments are parallel and cannot intersect. The problem of finding a maximum matching in a bipartite graph is known as the "marriage problem"; one can be found in $O(V^{1/2}E)$ time, where V is the number of vertices and E the number of edges of the bipartite graph (Even 1979). For a grid of n segments, $V = n$

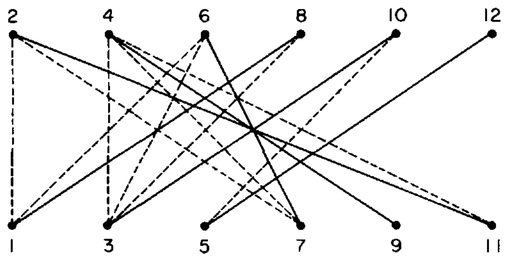


Fig. 9.15. The intersection graph for the grid in Fig. 9.14. The 6 guards are shown as solid lines.

and $E = O(n^2)$, so both the general algorithm and the bipartite algorithm lead to $O(n^{2.5})$ worst-case complexity.

For complete coverage of a grid, each line segment requires a guard, and since only two line segments cross at an intersection, a guard can cover at most two segments. Thus for a grid of n segments, at least $\lceil n/2 \rceil$ guards are necessary. If the intersection graph has a *perfect matching,* a matching of size $\lceil n/2 \rceil$, then $\lceil n/2 \rceil$ guards suffice: place a guard at each intersection of the grid corresponding to an edge of the matching. If a maximum matching has size m, then placing guards at the corresponding intersections covers $2m$ segments. The remaining $n - 2m$ segments can be covered with one guard each, resulting in total coverage by $m + n - 2m = n - m$ guards. Indeed this is a minimal cover: if fewer than $n - m$ guards suffice, more than m guards must cover two segments each that are not covered by any other guards, yielding a matching of size larger than m.

This argument establishes the following theorem.

THEOREM 9.3 [Ntafos 1985]. A minimum cover for a grid of n segments has $n - m$ guards, where m is the size of the maximum matching in the intersection graph of the segments, and may be found in $O(n^{2.5})$ time.

The dividing line between polynomial and NP-hard problems often seems to fall between problem parameters 2 and 3: 3SAT is NP-complete but 2SAT is polynomial, three-dimensional matching is NP-complete but graph matching is polynomial, vertex cover in graphs of degree at most 3 is NP-complete but polynomial if the degrees are at most 2. And Ntafos showed that minimal coverage of three-dimensional grids is NP-complete in contrast to the above theorem for two-dimensional grids. We now turn to this result.

The proof is by reduction from the vertex cover problem in graphs of degree at most three, a known NP-complete problem (Garey and Johnson 1979).

Vertex Cover

INSTANCE. A graph $G = (V, E)$ with all nodes of degree three or less; a positive integer $K < |V|$.

QUESTION. Is there a vertex cover for G of size at most K? That is, is there a set of vertices C of size K or less such that each edge of G is incident on at least one vertex of C?

The goal is to construct a three-dimensional grid P such that there is a cover by less than or equal to g guards (the value of g will be specified later) iff there is a vertex cover of G with less than or equal to K vertices. The basic idea is simple. Label the vertices of G $1, 2, \ldots, n = |V|$. The vertices of G are assigned to lattice points along the diagonal line through $(0, 0, 0)$ and $(1, 1, 1)$: each vertex labeled i is assigned the lattice point $v_i = (3i, 3i, 3i)$. Edges of G are represented by grid paths between the lattice points corresponding to the endpoint vertices of the edges. Because each vertex of G is of degree 3 or less, each incident edge can be assigned one of the three orthogonal directions without conflict.

We now describe the construction of the grid P from the given graph G. Suppose all edges incident to vertices $1, \ldots, i-1$ of G have been assigned paths in P, and consider an edge (i, j) of G, with $i<j$. We will use the convention that paths from v_i to v_j with $i<j$ will leave v_i along the positive rays $+x$, $+y$, and $+z$, and approach v_j along the negative rays $-x$, $-y$, and $-z$. Because vertex i has at most degree three, one of the three rays in the directions $+x$, $+y$, or $+z$ emanating from $(3i, 3i, 3i)$ must contain no segments of P. Assume that the $+x$ ray is unoccupied. Then connect $v_i=(3i, 3i, 3i)$ to $v_j=(3j, 3j, 3j)$ with these three grid segments, as long as they do not overlap with any previously constructed segments of P:

$$\begin{aligned}
&(3i, 3i, 3i) \rightarrow (3j, 3i, 3i),\\
&(3j, 3i, 3i) \rightarrow (3j, 3j, 3i),\\
&(3j, 3j, 3i) \rightarrow (3j, 3j, 3j).
\end{aligned}$$

The path moves in the $+x$, $+y$, and $+z$ directions in sequence, as illustrated in Fig. 9.16a. If the $+x$ direction is occupied but $+y$ is free at v_i, then the path moves $+y$, $+z$, and $+x$ in sequence. If only the $+z$ ray is unoccupied at v_i, the path follows $+z$, $+x$, and $+y$ to reach v_j. These alternative paths are illustrated in Figs. 9.16b and 9.16c. Note that each of the three alternative paths lies in a distinct plane containing v_i and a distinct plane containing v_j.

Now assume the attempted connection from v_i to v_j overlaps a segment of P containing v_j, due to an earlier connection from v_k to v_j, $k<i<j$. Let the overlap occur on the $-z$ ray from v_j. Then at least one of the rays $-x$ or $-y$ from v_j must be unoccupied. If the $-x$ ray is unoccupied, modify the (v_k, v_j) path as shown in Fig. 9.17a; if the $-y$ ray is unoccupied, modify as shown in Fig. 9.17b. In both cases the overlap on $-z$ is avoided, and the (v_k, v_j) path approaches v_j along the unoccupied ray. Note that the bent path now consists of nine edges. Similar modifications are made if the path overlaps along the $-x$ or $-y$ rays at v_j. Because v_j can have at most three incident paths in P, one of the rays $-z$, $-y$, $-x$ is always unoccupied when a connection is made, so overlap can always be avoided. It may be that a path that is already bent to avoid overlap, will have to bend again to avoid

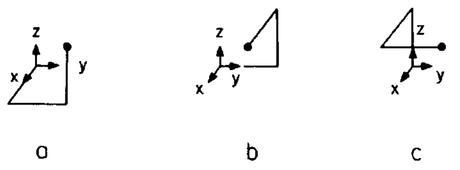


Fig. 9.16. Three paths from v_i (the origin) to v_j.

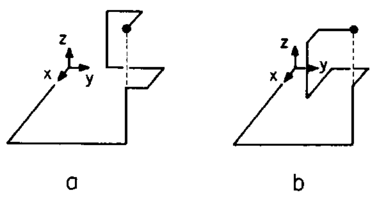


Fig. 9.17. Bending a path to avoid overlap on the $-z$ ray, to enter along the $-x$ ray (a) or the $-y$ ray (b).

overlap with the third incident edge, which would change the 9 segment path into one of 15 segments.[4]

After complete construction, the e edges of G will be embedded as e_3 paths of 3 segments each (when no overlap is encountered), e_9 paths of 9 segments each (when overlap forces bending as in Fig. 9.17), and e_{15} paths of 15 segments each (when overlap occurs twice), where $e = e_3 + e_9 + e_{15}$.

LEMMA 9.8. There is a vertex cover of size K of G iff there is a cover of the grid P whose construction is described above with $g = K + e_3 + 4e_9 + 7e_{15}$ guards.

Proof. Suppose there is a vertex cover C of G of size K. Then assign a guard to each intersection in the grid P that corresponds to a vertex in C. Let p_3 be a 3-segment path in P between v_i and v_j. Then because C is a vertex cover, at least one of v_i or v_j is assigned a guard; therefore one additional guard suffices to cover p_3. Let p_9 be a 9-segment path in P between v_i and v_j. Again one of v_i or v_j must be assigned a guard, so that p_9 can be covered with four additional guards, one on every other corner. Similar reasoning show that a 15 segment path requires seven additional guards. The result is complete coverage by $K + e_3 + 4e_9 + 7e_{15}$ guards.

Suppose P may be covered by $K + e_3 + 4e_9 + 7e_{15}$ guards. Each 3-segment path requires one guard on an internal corner—that is, not at a grid point corresponding to some vertex of G. Similarly each 9-segment path requires four guards on internal corners, and each 15-segment path requires seven such guards. This accounts for $e_3 + 4e_9 + 7e_{15}$ guards. Thus at most K guards may be located at grid points corresponding to vertices of G. Suppose the guards assigned to these grid points do *not* correspond to a vertex cover of G. Then some edge (i, j) of G is not incident to a vertex of the cover, which means that its associated path in P does not have a guard at v_i nor at v_j. But then, if the path is a 3-segment path, the one guard at an internal corner does not suffice to cover the path, since this guard can only cover two of the three segments. Similarly, if the path is a 9-segment or 15-segment path, the four or seven guards at internal vertices cover at most 8 or 14 of the 9 or 15 segments, respectively. Thus if the K guards do not correspond to a vertex cover, coverage of the grid P cannot be achieved with $K + e_3 + 4e_9 + 7e_{15}$

4. I have modified Ntafos's argument somewhat.

guards. Thus those K guards must correspond to a vertex cover of at most size K. □

THEOREM 9.4 [Ntafos 1985]. Minimal guard coverage of a three-dimensional grid is NP-complete.

Proof. The problem is clearly in NP, as guards need only be located at corners or junctions, and an optimal placement may be guessed and checked in polynomial time. The reduction from Vertex Cover establishes that the problem is NP-complete. □

9.5. PARTITIONS WITHOUT STEINER POINTS

The previous section showed how a version of the minimal guard coverage problem that is solvable in polynomial time can be obtained by severely restricting the class of polygons covered. Another problem solvable in polynomial time can be obtained by restricting the pieces of the decomposition rather than restricting the shape of the polygon being decomposed. Keil showed that the problem of finding a minimal *partition* of an arbitrary simple polygon into star pieces *without* Steiner points can be solved in $O(n^7 \log n)$ time (Keil 1984, 1985b). We showed in Section 9.1 that the number of pieces in a minimal partition into stars can be much larger than the number of guards required for coverage: a partition is quite different from a cover. But it is precisely the restriction to partitions that permits a polynomial algorithm to find the minimum. Consider a diagonal d that lies on the boundary of a star in a minimal star partition of P. P is partitioned into two polygons by d, P_1, and P_2. The crucial observation is that the minimum partition of P is the union of the minimal partitions of P_1 and P_2. Note that this additive property does not hold for covers as d might be overlapped by a piece in a cover. But the fact that the pieces of a partition do not overlap permits a dynamic programming algorithm to build up a minimal partition for P from minimal partitions of subpolygons in P. We now proceed with the details, which are a bit complicated.

Let the vertices of the polygon P to be partitioned be labeled from 1 to n counterclockwise. Let P_{ij} be the subpolygon composed of the boundary of P from i to j, and the diagonal (j, i). P_{ij} is defined only if $i < j - 1$ and i can see j. A minimum partition of P_{ij} will be constructed from minimal partitions of P_{im} and P_{mj} for $i < m < j$. The diagonal (i, j) is called the *base* of P_{ij}, and for any minimal partition M of P_{ij}, $S_{ij}(M)$, the *base star* of M, is the star polygon in M that includes (i, j). Finally, T_{imj} is the triangle whose base is (i, j) and whose apex is m. These definitions are illustrated in Fig. 9.18. The basic idea of the dynamic programming algorithm is to build a minimal partition M of P_{ij} by merging T_{imj} with the minimal partitions A of P_{im} and B of P_{mj}. If $S_{im}(A) \cup T_{imj}$ is a star polygon, then T_{imj} is said to *single merge* with partition A; if $S_{im}(A) \cup T_{imj} \cup S_{mj}(B)$ is a star, then T_{imj} is said to *double merge* with partitions A and B.

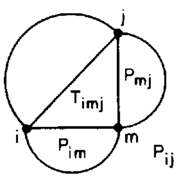


Fig. 9.18. The subpolygon P_{ij} and associated regions.

We now concentrate on defining the *states* of the dynamic programming algorithm. First observe that a state cannot be represented by just one minimal partition of a subpolygon. Consider Fig. 9.19. P_{im} is minimally partitioned into two pieces with either diagonal a or b (or others). If diagonal a is used to partition P_{im}, then T_{imj} can single merge with the base star of the partition, while if b is used, the merge produces a non-star polygon. This suggests that each minimal partition of a subpolygon must be a separate state. There are two difficulties with this approach. The first is that there can be an exponential number of distinct minimal partitions: the polygon shown in Fig. 9.20 has 2^{k-1} distinct partitions into k stars, where $k = \Theta(n)$ is the number of steps of the staircase, since there are two independent choices for the diagonal separating adjacent stars. In the figure, $k = 4$. The second difficulty is that sometimes a minimal partition of P_{ij} cannot be constructed from single or double merges. Consider Fig. 9.21. Here P_{ij} is a star, but since the kernel of P_{ij} lies inside every T_{imj} for $i < m < j$, the subpolygons P_{im} and P_{mj} are not both stars. Thus their minimal partitions will contain more than one piece, and simple merges with T_{imj} will never result in the true minimal partition of P_{ij}.

Keil solved these problems by introducing the notion of a pseudo-star. A *pseudo-star polygon* PS_{ij} based on (i, j) is a polygon such that there exists a point x in P but not in PS_{ij} such that x can see all of PS_{ij} *through* (i, j). Thus PS_{ij} is a portion of a star polygon whose kernel lies on the other side of (i, j). For example, P_{im} in Fig. 9.21 is a pseudo-star based on (i, m). Extending minimal partitions to permit a pseudo-star polygon on the base solves the second difficulty mentioned above.

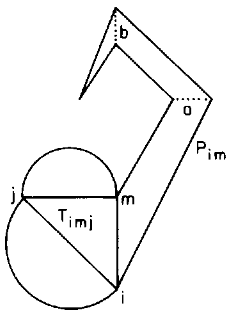


Fig. 9.19. T_{imj} cannot merge with every minimal partition of P_{im}.

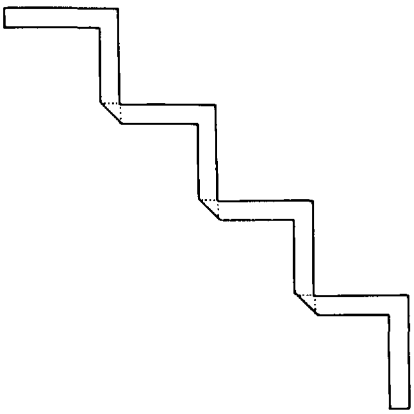


Fig. 9.20. A polygon with 8 distinct partitions into 4 stars.

We now reconsider the first difficulty, the exponentially many partitions of a subpolygon. Although it is not possible to save just one minimal partition for a subpolygon, as Fig. 9.19 showed, only variations in the base star (or pseudo-star) are relevant for merging, as merging only occurs at the base. Thus we need to identify all possible base pseudo-stars. This goal motivates the following definition.

Let L be the set of all maximal line segments internal to P determined by two vertices of P, at least one of which is reflex. Define K to be the vertices of P unioned with the set of all intersection points between the lines of L with each other and with the edges of P. L has size $O(rn)$ for a polygon of n vertices r of which are reflex, and therefore K has size $O(r^2n^2)$. This set K reduces the pseudo-star kernel points to a polynomial-size set of possibilities.

LEMMA 9.9. The kernel of any base star polygon of a minimal partition of a subpolygon P_{ij} contains a point in K.

Proof. Let M be a minimal partition of P_{ij}, and let $S_{ij}(M)$ be its base star. If S_{ij} is convex, then vertex i is in its kernel and in K. If S_{ij} is not convex, then its kernel is a proper subset of S_{ij}. The kernel is the intersection of all

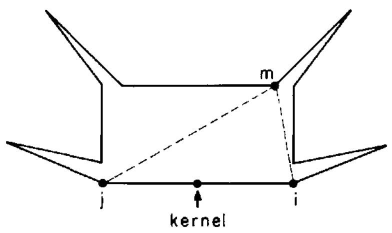


Fig. 9.21. A simple merge of T_{imj} and minimal partitions of the remaining subpolygons will not yield a single star.

the half planes defined by edges of S_{ij}. At least one of the half-planes that form part of the boundary of the kernel must be determined by an edge with a reflex vertex as endpoint; for edges with two convex endpoints extend exterior rather than interior to the polygon. Thus at least one edge of the kernel is determined by a line segment l in L, the set of line segments defined above. If the intersection of l with P lies in the kernel, then the lemma is established since this point is in K. If not, then another line l' containing an edge of the kernel must intersect l. Since l' must be defined by an edge with a reflex endpoint, the intersection of l and l' is in K. □

Since the kernel of any base star must contain a point in K, we only need consider pseudo-stars visible from the points in K. Let M_x be a minimal partition of P_{ij} with a base pseudo-star PS_{ij} visible from x. Although there may be exponentially many such M_x for a given x, we need only retain one representative. This representative is a state for the dynamic programming algorithm. Although we must store M_x for all $x \in K$, K is of polynomial size. Thus the exponential explosion has been circumvented.

We may finally specify the algorithm.

Preprocessing

(1) Compute the set K. $[O(r^2n^2)]$
(2) Compute the visibility graph of P. $[O(n^2)]$
(3) Construct the subpolygons and sort by number of vertices. $[O(n^2 \log n)]$
(4) Form a list of all base triangles T_{imj}. $[O(n^3)]$

Dynamic Programming

```
for each $P_{ij}$ (in order of increasing size) do
  for each $T_{imj}$ do $[O(n^3)]$
    for each $x \in K$ do $[O(r^2n^2)]$
      double merge:
        if both $M_x$ of $P_{im}$ and $M_x$ of $P_{mj}$ exist
        then $T_{imj}$ can merge with both
      single merge:
        else if $M_x$ of $P_{im}$ exists and $j$ sees $x$
        then $T_{imj}$ can merge with $P_{im}$
      no merge:
        else if $x \in T_{imj}$ or $x$ sees $T_{imj}$ through $(i, j)$
        then candidate $M_x$ for $P_{ij}$
```

A smallest partition of P_{1n}, the last subpolygon to be considered, is a minimal partition for P. A careful accounting shows that the algorithm runs in $O(r^2n^5 \log n) = O(n^7 \log n)$ time (Keil 1985b). Because of this high complexity, it is difficult to illustrate the algorithm by example. We will settle for showing just the final step on a small example.

Let P be the 11 vertex polygon shown in Fig. 9.22, and consider the status of the algorithm when $i = 1$, $j = 11$, and $m = 5$. P_{15} is a star, and so its

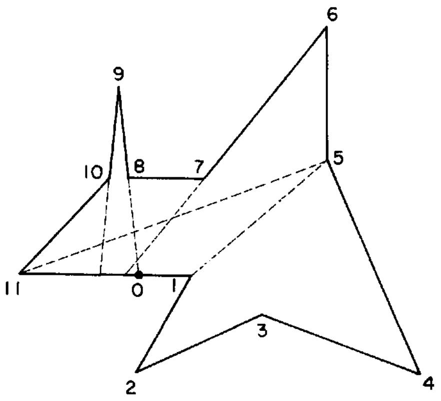


Fig. 9.22. An example of the star partitioning algorithm: the polygon may be partitioned into two stars.

minimal partition is $A = \{(1, 2, 3, 4, 5)\}$. Several pseudo-star partitions would have been computed for P_{15} by the algorithm, including $A_1 = \{(1, 2, 3, 4, 5)\}$, since P_{15} is visible from 1. $P_{5,11}$ is not a star. Its only minimal partition has three pieces: $B = \{(5, 6, 7), (7, 8, 9, 10, 11), (5, 7, 11)\}$. However, it has several more efficient pseudo-star partitions, including $B_0 = \{(5, 6, 7, 8, 9, 10, 11)\}$, where 0 is the intersection of the line containing edge $(8, 9)$ with the boundary of P, since all of $P_{5,11}$ is visible from 0 through $(5, 11)$. Many other pseudo-star partitions exist for $P_{5,11}$, including $B_1 = \{(5, 6, 7, 8, 11), (8, 9, 10, 11)\}$, where all of $(5, 6, 7, 8, 11)$ is visible from 1 through $(5, 11)$. Now consider the attempt to merge $T_{1,5,11}$ with the partitions of $P_{1,5}$ and $P_{5,11}$. First let $x = 1 \in K$. Then since both A_1 and B_1 exist, a double merge is possible. The result is the partition $M_1 = \{(1, 2, 3, 4, 5, 6, 7, 8, 11)(8, 9, 10, 11)\}$. Second, let $x = 0 \in K$. Although B_0 exists, A_0 does not, since no pseudo-star based on $(1, 5)$ is visible from 0. Thus no double merge is possible. However, since B_0 exists and 1 can see 0, a single merge is possible, resulting in $M_0 = \{(1, 5, 6, 7, 8, 9, 10, 11)(1, 2, 3, 4, 5)\}$. Both M_0 and M_1 are minimal partitions of P.

Although we have emphasized that the minimum partition problem is polynomial because it admits a dynamic programming algorithm, whereas the NP-complete minimum cover problem does not, Lingas showed that the proof in Section 9.2 can be modified to establish that minimum partition is NP-complete for polygons with holes (Lingas 1982b). He observed that the stars in the decomposition need only overlap at the crossovers, and that Lichtenstein established that planar 3SAT is NP-complete (Lingas 1982a). Thus Keil's dynamic programming approach cannot work for polygons with holes. Intuitively this is because a diagonal in a multiply-connected polygon P does not necessarily cut it into two pieces, and therefore merging is not confined to one base edge, but must in the worst case be considered along the entire boundary of a subpolygon.

9.6. DISCUSSION

The results discussed in this chapter are summarized, together with two results not yet discussed, in Table 9.1. The results on monotone orthogonal polygons were obtained by Keil (cover) and Liu and Ntafos (partition) (Keil 1985a; Liu and Ntafos 1985). Several interesting open questions remain:

(1) Can a variant of Keil's dynamic programming approach be used to find star partitions permitting Steiner points? Chazelle was able to achieve $O(n^3)$ for minimum *convex* partition with Steiner points via a very complex dynamic programming algorithm (Chazelle 1980), but star partitions seem even more complicated.

(2) What are the complexities of the various problems when restricted to orthogonal polygons? As the table indicates, the first inroads have already been made for monotone orthogonal polygons. It does not seem to be straightforward to extend these results to general orthogonal polygons, however.

(3) Are there other natural restrictions on the pieces that result in polynomially solvable problems? Aggarwal *et al.* investigated star covers where the boundary of the pieces are formed by extensions of the edges of the polygon (Aggarwal *et al.* 1985). They claim that Lee and Lin's algorithm can be modified to establish that this restricted problem is also NP-complete.

(4) Given that most of the interesting problems seem to be intractable, it is natural to seek approximation algorithms that achieve decompositions with, say, no more than a constant times the optimal number of pieces. Such approximation algorithms are just beginning to be explored.

Table 9.1

Decomposition→	Cover		Partition	
	w. Steiner	w/o Steiner	w. Steiner	w/o Steiner
simple polygons	NP-hard	NP-complete	?	$O(n^7 \log n)$
polygons with holes	NP-hard	NP-complete	NP-hard	NP-complete
monotone orthogonal polygons	$O(n^2)$		$O(n)$	
two-dimensional grids	$O(n^{2.5})$		?	
three-dimensional grids	NP-complete		NP-complete	