The Computational Complexity of Cartographic Label Placement

Joe Marks and Stuart Shieber

March 22, 1991
Revised April 11, 1993

Abstract

We examine the computational complexity of cartographic label placement, a problem derived from the cartographer's task of placing text labels adjacent to map features in such a way as to minimize overlaps with other labels and map features. Cartographic label placement is one of the most time-consuming tasks in the production of maps. Consequently, several attempts have been made to automate the label-placement task for some or all classes of cartographic features (punctual, linear, or areal features), but all previously published algorithms for the most basic task—point-feature-label placement—either exhibit worst-case exponential time complexity, or incorporate incomplete heuristics that may fail to find an admissible labeling even when one exists. The computational complexity of label placement is therefore a matter of practical significance in automated cartography. We show that admissible label placement is NP-complete, even for very simple versions of the problem. Thus, no polynomial time algorithm exists unless $P = NP$. Similarly, we show that optimal label placement can be solved in polynomial time if and only if $P = NP$, and this result holds even if we require only approximately optimal placements. The results are especially interesting because cartographic label placement is one of the few combinatorial problems that remains NP-hard even under a geometric (Euclidean) interpretation. The results are of broader practical significance, as they also apply to point-feature labeling in non-cartographic displays, e.g., the labeling of points in a scatter plot.

Key words: Automated cartography, computational complexity, computational geometry, computer graphics, heuristic methods, label placement, NP-completeness.

Contents

1 Introduction and Overview[1]

Text labels are an essential part of almost all maps: labels identify point features like cities and towns, linear features like roads and rivers, and areal features such as parks and lakes (see Figure 1 for an example of a densely labeled map). Before the widespread use of computers in cartography, lettering tasks (i.e., all tasks involving labels, explanations, and numbers) sometimes accounted for more than 50 percent of the production time for maps [15]. Label placement is also a significant problem in the production of other kinds of graphical displays. For example, labeling the points in a scatter plot is another instance of the problem discussed here. Cartographic labeling is a skilled task, requiring the cartographer to consider many competing criteria when choosing the location, orientation, shape, and typography of text labels. In a classic paper on cartographic label placement, Imhof illustrates many of the concerns affecting label placement with over 100 examples of good and bad labeling decisions [6].[2] Two concerns stand out as being of particular importance: the first concern is the degree to which labels overlap and obscure cartographic features (including other labels); and the second concern is the degree to which labels are unambiguously and clearly associated with the features they identify. The latter concern is illustrated in Figure 2, which shows relative rankings for various label placements relative to the point features they identify.

Informally, the setup of a point-feature-label placement problem involves the positioning of labels for point features, where a label can be placed in any one of a set of locations adjacent to the point it labels. A problem instance is given by a set of points, with their positions in the plane and their label size, along with a set of constraints on the labeling. The constraints may be binary (for instance, a prohibition against a label overlapping a point, another label, or some other cartographic feature) or continuous (for instance, giving preferences among label positions or penalties for degree of overlap of two labels). As a minimum requirement, however, the constraints must provide for some penalty for labels overlapping other points or labels. Schematically, we disprefer situations like the following two:

[1]The research reported in this paper was funded under a contract with U S West Advanced Technologies. We would like to thank Stanley Chen, Jon Christensen, Scott Decatur, Steve Feiner, Harry Lewis, Victor Milenkovic, Christos Papadimitriou, Fernando Pereira, Sivan Toledo, and Mark Tuttle for helpful discussions on the topic of this paper.

[2]An English translation of this paper was published subsequently [7].

Figure 1: A sample map [Rand-McNally Company, *Universal World Atlas* (third edition); reproduced by permission of the publisher]

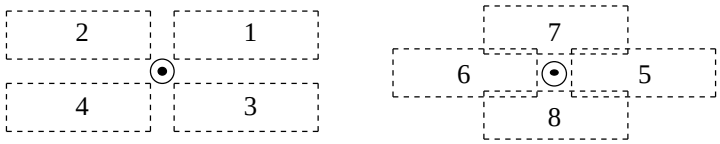
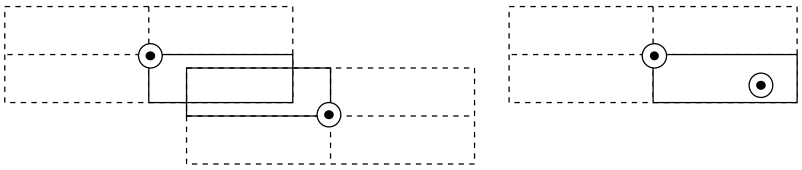


Figure 2: Desirability of different point-feature-label placements



(The possible label regions are given by dashed lines, the actual label position by a solid line.) In the first case, a label overlaps another label; in the second, a label overlaps another point.

The problem of label placement is to determine an optimal labeling of the points, or, in the case where all the constraints are binary, an admissible labeling of the points. Even more simply we might want to determine whether such an admissible labeling exists. The size of a problem is conveniently given by the number of points to be labeled.

Given the difficulty of label placement, cartographers have looked for ways to automate the label-placement task. Although the labeling of linear and areal features are by no means easy tasks, the most difficult task is that of label placement for point features. We can use the two most important of Imhof's concerns to fashion a formal statement of the optimal label-placement problem for point features as a purely combinatorial problem. Suppose we are given a set of points P to label, each with a position in the plane and a label size, together with a fixed set N of possible label positions. For each point $p \in P$ and label position $n \in N$, it is straightforward to determine the region of the plane covered by the label for p at position n. We will notate that region $region(p, n)$. In general, this can be computed from the point's position and label size and the relative positioning of the label position. A diagram such as Figure 2 can be thought of as a specification of the $region$ function.

We will assume that there are costs or penalties associated with violating certain standard criteria for label placement. First, penalties for the overlapping of a point's label by some other cartographic feature (including other points) is specified as a function $c_{pl} : P \times N \to \mathbb{Z}^+$. We require that this cost function be consistent with the placement of the points in the problem, that is, that

$$c_{pl}(p, n) > 0 \quad \text{if there is a point } p' \in P \text{ such that } p' \in region(p, n) \qquad (1)$$

(Here, we are using set notation applied to points and regions in an obvious way, and will continue to do so.) Other than this constraint, the c_{pl} cost

function is free to assign any costs to particular point/position combinations. In particular, any *a priori* preferences among relative label positions can be folded into this cost function.

Second, overlapping labels must be penalized. We require a cost function $c_{ll} : P \times N \times P \times N \to \mathbb{Z}^+$ that must penalize overlapping labels, that is,

$$c_{ll}(p, n, p', n') > 0 \text{ if } region(p, n) \cap region(p', n') \neq \emptyset \quad . \tag{2}$$

Having defined the characteristics of the cost functions, we can define label-placement problems more formally. We first consider label placement that optimizes the cost functions.

Optimal Point-Feature-Label Placement (OLP)
Instance: A set P of point features each with a position in $\mathbb{R}^2$ and a label size, each of which can be labeled in one of a set N of relative positions (for example, the eight relative positions in Figure 2); cost functions c_{pl} and c_{ll} satisfying (1) and (2), respectively.
Question: What label placement $g : P \to N$ minimizes the cost function

$$\sum_{p \in P} c_{pl}(p, g(p)) + \sum_{p,q \in P} c_{ll}(p, g(p), q, g(q)) \quad ?$$

Although the description is geometrically evocative, the problem as stated is purely combinatorial, in that nothing requires us to interpret the $region(p, n)$ function in accord with any particular geometry, although we will in general interpret it in the standard Euclidean manner.

By further refining c_{pl} and c_{ll}, we can instantiate this general problem to form a specific variant that ignores issues of preference among label positions and optimization of degrees of overlap and that is stated as a decision problem so as to be amenable to NP-hardness arguments. We do so by specifying further constraints on c_{pl} and c_{ll}, which now take their values from the set $\{0, 1\}$:

$$c_{pl}(p, n) = 1 \quad \text{if there is a point } p' \in P \text{ such that } p' \in region(p, n) \quad . \tag{3}$$

$$c_{ll}(p, n, p', n') = 1 \text{ if } region(p, n) \cap region(p', n') \neq \emptyset \quad . \tag{4}$$

Because solution of the problem now depends only on whether there are any overlaps, and not on their number or degree, the problem involves finding a merely admissible (non-overlapping) labeling, as opposed to an optimal one.

Admissible Point-Feature-Label Placement (ALP)
Instance: A set P of point features each with a position in $\mathbb{R}^2$ and a label size, each of which can be labeled in one of a set N of relative positions; cost functions c_{pl} and c_{ll} satisfying (3) and (4), respectively.
Question: Is there a label placement $g : P \to N$ such that

$$\sum_{p \in P} c_{pl}(p, g(p)) + \sum_{p,q \in P} c_{ll}(p, g(p), q, g(q)) = 0 \qquad ?$$

All previously published algorithms for various variants of OLP are heuristic in nature. (See Section 2.) Some of these algorithms do not guarantee to place labels for all point features, even though some such labeling exists. Of those algorithms that do guarantee to find a labeling if it exists, all exhibit exponential worst-case time complexity. The computational complexity of point-feature-label placement is therefore a matter of practical significance in automated cartography. We show that OLP is solvable in polynomial time if and only if $P = NP$ by showing that a particular instance of ALP is NP-complete. We then generalize the results to include a broader class of label-placement algorithms, including algorithms that generate approximately optimal labelings.

Of particular interest is the fact that the instance of ALP that we choose respects the geometric interpretation of the problem. Thus cartographic label placement is one of the few purely combinatorial NP-complete problems that does not get easier by restricting it to its geometric variant. (Another notable exception is the Traveling Salesman Problem. [10])

2 Previous Research on Cartographic Label Placement

One of the earliest attempts at automating label placement was due to Yoeli [15]. Yoeli's algorithm is basically a deterministic greedy algorithm for label placement; it considers each point feature in turn, placing labels at the best position that does not result in an overlap. (The ranking of relative positions used by Yoeli is similar to that of Figure 2.) If a point feature cannot be labeled, it must be placed manually.

Jones [8] presents a nondeterministic greedy algorithm. Rather than requiring manual intervention when a label cannot be placed, Jones's algorithm will backtrack recursively in its search for an admissible labeling.

When backtracking fails to yield an admissible labeling, the algorithm starts over with lower tolerances for overlap detection. Deletion of point features is the option of last resort if an admissible labeling cannot be found. Jones's algorithm includes a number of useful efficiency measures, such as an initial phase in which groups of point features that can be labeled independently are identified.

The identification of independent point-feature groups was suggested originally by Ahn and Freeman [1]. Their algorithm is based on the idea of state-space search: in the initial state no point features are labeled, and in the goal state all features are labeled. They explore the state space using a heuristic search method that they claim is similar to the A^* algorithm [9], though few details are given. If a state is encountered in which no additional features can be labeled, backtracking is performed.

Hirsch's heuristic algorithm is based on the notion of an "overlap vector" [5]. In Hirsch's algorithm all labels are placed initially in their most preferred positions. All overlaps are then detected. For each overlap a vector is computed that indicates a direction of movement that would eliminate the overlap. All overlap vectors for a label are summed to give an accumulated overlap vector. Hirsch suggests a number of ways in which accumulated overlap vectors can be used to modify label placement. For example, labels can be moved to the nearest preferred position in the direction of the accumulated vector. The process of overlap detection, vector calculation, and label movement is repeated an arbitrary number of times.

The algorithms discussed above may not guarantee to find an admissible labeling when one exists. Zoraster's algorithm [16], on the other hand, will find an optimal labeling if one exists, but has exponential time complexity in the worst case. Zoraster formulates the label-placement problem as the following 0-1 integer programming problem:

> *Variables:* One label-placement variable $x_{n,p}$ for each point feature ($1 \leq p \leq |P|$) and relative label position ($1 \leq n \leq |N|$). Each $x_{n,p}$ is constrained to have value 0 or 1 (a 1 indicates that the label for feature p is to be placed in position n).
>
> *Constraints:* To ensure exactly one position per label: $\sum_{n=1}^{|N|} x_{n,p} = 1$ for each p, $1 \leq p \leq |P|$. To avoid overlapping labels: $x_{n,p} + x_{n',p'} \leq 1$ for each pair of label positions, $x_{n,p}$ and $x_{n',p'}$, that overlap.

Cost Function: $\sum_{p=1}^{|P|} \sum_{n=1}^{|N|} w_{n,p} x_{n,p}$, where $w_{n,p}$ is a weighting that reflects the desirability of the label position $x_{n,p}$.

In essence, Zoraster solves a version of OLP where

$$c_{pl}(p, n) = w_{p,n}$$

and

$$c_{ll}(p, n, p', n') = \begin{cases} 1 & \text{if } region(p, n) \cap region(p', n') \neq \emptyset \\ 0 & \text{otherwise} \end{cases} .$$

Various methods are available for the approximate solution of 0-1 integer-programming problems, and Zoraster mentions several as being of potential use for this problem.

The less-demanding—but not trivial—problems of label placement for linear and areal features have received relatively little attention. The integer-programming approach devised by Zoraster can also accommodate labels for a special kind of linear feature (a seismic shot line) that appears on oil-industry maps [16]. A comparison of simple mathematical methods for spot-symbol placement inside areal features is presented in [2]: the methods described there could be applied to the placement of areal-feature labels. Van Roessel [14] describes an algorithm for identifying candidate rectangular zones for labels within polygonal areal features. Ahn and Freeman [1] also address areal-feature and linear-feature labeling. Their approach to areal features is unique in that it takes into account the shape of the feature and tries to conform the label to that shape. The complexity of some related computational-geometry problems has also been analyzed: Chazelle has considered the problem of polygon containment (some versions of the areal-feature-label-placement problem can be stated in terms of placing one polygon within another) [3], and Souvaine and Van Wyk have analyzed the difficulty of arranging the labeled sectors of a pie chart [12].

3 The Simple Label-Placement Problem

The problem that we begin with is a very simple variant of the ALP problem presented above, namely, one that allows only four relative label positions (as depicted in the left half of Figure 2), fixed-size labels, and penalties given exclusively by overlap with other point features and labels. Since we assume that all labels are of a given fixed size, the label size will be a parameter of

the problem. A particular pair of cost functions will be specified that assign penalties only for overlap of a label with another label, a labeled point, or one of a set of unlabeled points also given as a parameter of the problem. The use of a set of unlabeled points lets us use a uniform cost function c_{ll} for all instances of the simple label-placement problem. In particular, only those binary cost functions that can be specified in terms of overlapping of a set of points are allowed. The more general problems are not subject to this implicit restriction, so that it would be possible to specify a cost function that disallowed some label position for one point but allowed some label position for another point, even though the two label regions were coextensive in the plane. This kind of cost function is eliminated in the simpler setup.

Simple Admissible Point-Feature-Label Placement (SLP)
Instance: A set P of point features each with a position in $\mathbb{R}^2$, and each of which can be labeled in one of the 4 relative positions given in Figure 2(a); a set U of point features each with a position in $\mathbb{R}^2$ that are not to be labeled; numbers X and Y that give the fixed horizontal and vertical extent of a label.
Question: Is there a label placement $g : P \to N$ such that

$$\sum_{p \in P} c_{pl}(p, g(p)) + \sum_{p,q \in P} c_{ll}(p, g(p), q, g(q)) = 0$$

where

$$c_{pl}(p, n) = \begin{cases} 1 & \text{if there is a point } p' \in P \cup U \\ & \quad \text{such that } p' \in region(p, n) \\ 0 & \text{otherwise} \end{cases}$$

and

$$c_{ll}(p, n, p', n') = \begin{cases} 1 & \text{if } region(p, n) \cap region(p', n') \neq \emptyset \\ 0 & \text{otherwise} \quad ? \end{cases}$$

Any algorithm to solve Zoraster's problem can be used to solve SLP.[3] Similarly, Zoraster's problem can itself be reduced to ALP. Thus, a proof of the

[3] It is worth noting that although SLP reduces to Zoraster's problem, it is not necessarily the case that label placement for the four-position case reduces to label placement with other sets of label positions. That is, all instances of Zoraster's problem are not equally difficult. One need only observe that the one-position label-placement problem has an obvious n^2 algorithm; there is only a single layout to be checked. Thus, separate proofs are needed for, say, the six-position variant, or the continuous variant proposed by Hirsch [5]. Nonetheless, for many interesting cases of label placement, SLP will reduce appropriately. We return to this issue in Section 6.

NP-hardness of SLP constitutes a proof of the NP-hardness of Zoraster's problem and ALP. That these problems are in NP, hence NP-complete, is essentially trivial: we merely guess the correct layout from among the at most $|N|^{|P|}$ possible layouts and check that its cost is 0. This can clearly be done in polynomial time, as the cost is a function of pairs of points. In the next section, we prove the NP-hardness of SLP, and in Section 5 we use this result to prove relative upper and lower bounds on OLP, namely that OLP is solvable in polynomial time if and only if $P = NP$.

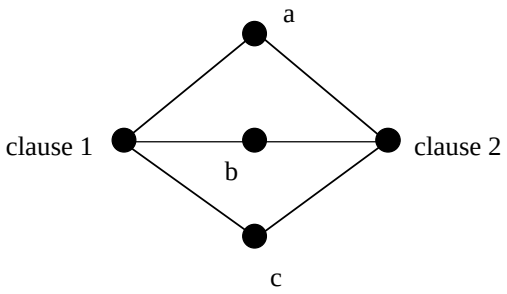
4 Reduction of Planar 3-SAT to Simple Label Placement

The NP-hardness of SLP is shown by a polynomial reduction of an NP-complete problem, PLANAR 3-SAT, to the SLP label-placement problem.

An instance of the PLANAR 3-SAT problem is given by a planar propositional formula F in 3-disjunctive normal form. A formula is planar if the following bipartite graph is planar: the graph (V, E) where V contains one node for each clause in F and one for each variable, and there is an edge in E connecting nodes c and x if and only if the clause corresponding to c contains a positive or negative occurrence of the variable x. For instance, (1) is a planar 3-DNF formula, whereas (2) is nonplanar.

1. $(a + b + c) * (\overline{a} + b + \overline{c})$
2. $(a + b + c) * (\overline{a} + b + \overline{c}) * (a + \overline{b} + \overline{c})$

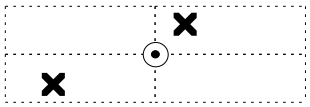
The bipartite graph for example (1) can be laid out as



but adding the vertex for the third clause generates the well-known nonplanar graph K_{33}. Thus, no planar graph exists for example (2).

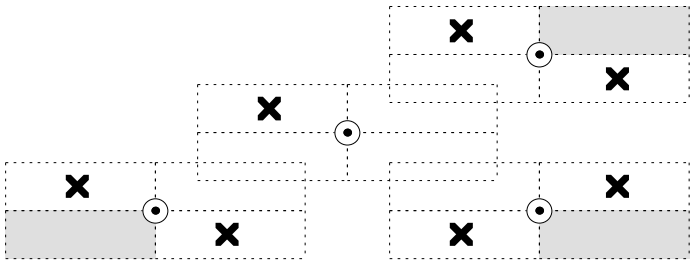
The planar 3-SAT problem—the problem of determining if a planar 3-DNF formula is satisfiable—is an NP-complete problem [4]. In order to reduce a planar 3-SAT problem to a label-placement problem, we provide a method for constructing from any planar 3-DNF formula sets of labeled and unlabeled points such that there is an admissible labeling of the points if and only if the formula is satisfiable. Further, the number of points is polynomial in the size of the planar 3-SAT problem, given as the product of the number of clauses $|C|$ and the number of variables $|U|$. We define σ to be $|U| + |C|$.

Before describing the construction, we show a few useful subconstructions that can be formed from points. First, we note that we can construct a point set with exactly two possible labelings:



Here, a labeled point (notated with its possible label regions as usual) and two unlabeled points (conventionally notated with an $\times$) are combined such that the only permissible labelings are the northwest and southeast.

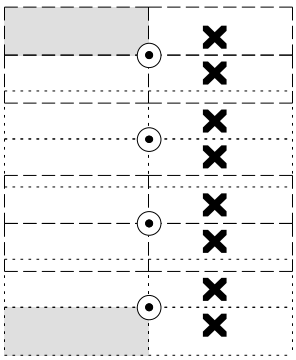
The following construction has the property that at least one of the shaded regions must be used to label its associated point, because if all three nonshaded regions are used for labeling the outside points, then the inner point has no region in which to place its label.



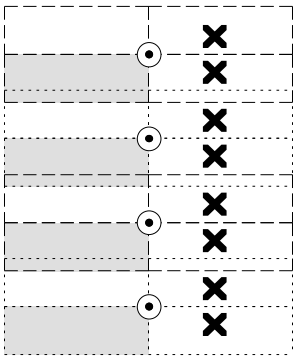
This construction will be used to encode a clause, and will be referred to as a *clause unit*. To satisfy a clause $(a + b + c)$, one of the three literals a, b, or c must be assigned the value $true$. The three outer points will correspond to the literals a, b, and c, and placing a label in the shaded region corresponds to selecting that literal to be true. Placing a label in the nonshaded region corresponds to a "don't care" condition. We are not requiring that literal

to be true; it may be true or false. Thus, this unit enforces the condition that at least one of the literals in the clause be true. We will call the literal that has its label in a shaded region the *selected literal*.

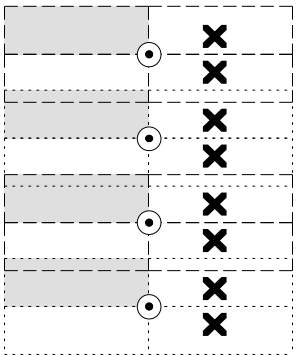
We can place instances of this construction in the plane, one for each clause. (This requires $O(|C|)$ points.) The only further requirement is that the label placements correspond to a consistent choice of values for the literals. In particular, we must guarantee that if a literal a in one clause is selected to be true, a literal $\overline{a}$ in another clause is not also selected. We know that the formula is planar, so that we can lay out the clause units corresponding to the positions of the clause vertices in the planar graph. For each variable, we will transmit to a central location the information about its selection status in each clause in which it occurs. Here, we will make use of a transmission line construction:



(We have used two different types of dashed lines to delineate the overlapping regions more clearly.) If a label from another point intrudes into the upper shaded region, then the label for the lower point must be placed in the lower shaded region. The label intrusion into the upper region is therefore "transmitted" to the lower region. The transmission line thus has two states, the selected state, in which the following labeling is forced:

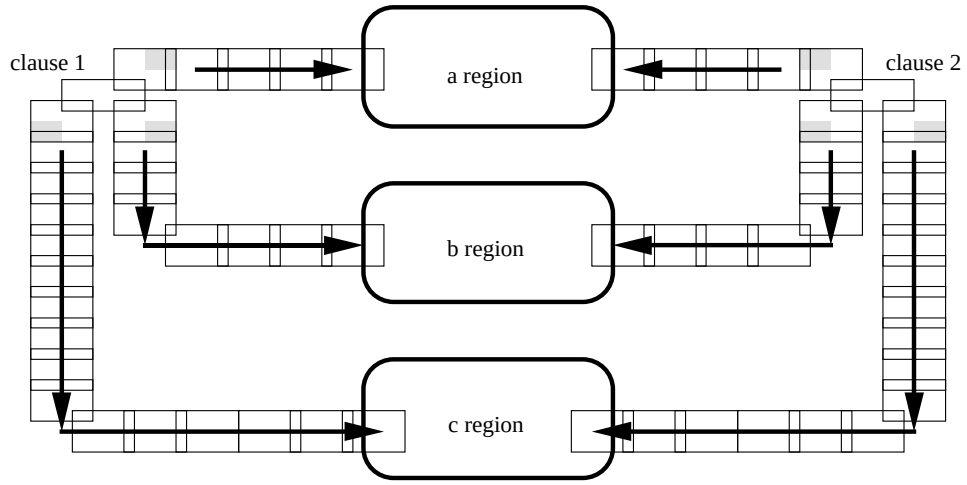


and an unselected state, in which the lowest label region is not forced to be used, as demonstrated by the following possible (but not forced) labeling:



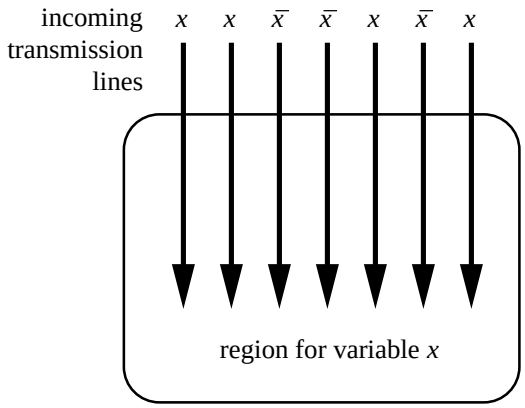
(The shaded regions correspond to the label placements.)

In this way, the information about which literals were selected can be transmitted from one place to another. Since we can construct both horizontal and vertical transmission lines, we can transmit this information from any place on the plane to any place else, so long as the transmission lines do not have to cross. In particular, since the formula is planar, we can transmit the information about literal selection from all literals of the same variable to a single region of the plane. For the graph for (1) above we have:

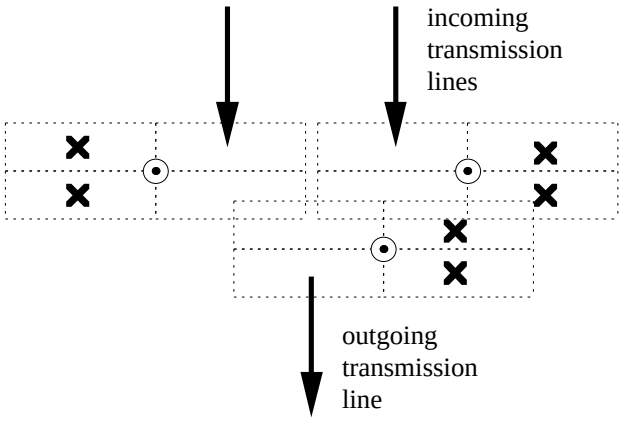


The method of Tammasia and Tollis [13] can be used to lay out a planar graph of V points along an integer grid of space $O(V) \times O(V)$ with at most a constant number of bends per line. Recall that the vertices in the planar graph correspond to the union of the clauses and variables. Thus, each point in the integer grid must have room for a clause unit or a variable region. The maximum width and height of a clause unit is constant. As we will see shortly, the maximum width and height of a variable region is $O(\sigma)$, so the integer grid must have an area of $O(\sigma^2) \times O(\sigma^2)$. Each transmission line can traverse the width of the grid at most a constant number of times (corresponding to the maximum number of bends) so the maximum number of points needed in all transmission lines is $O(\sigma^3)$.

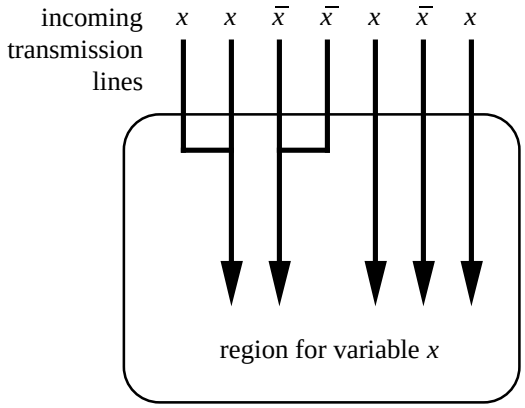
For each region of the plane corresponding to a variable, there will in general be several transmission lines coming in, some corresponding to positive occurrences of the variable, and some to negative occurrences. We must guarantee that a positive and negative occurrence cannot both be selected in two clause units. We do this by constructing a unit for the variable that enforces the constraint. Consider the general situation:



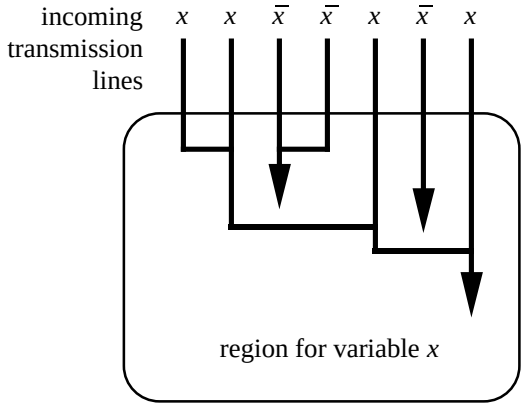
(Note that we can always arrange for the transmission lines to enter from above in the fashion given in the figure. The extra routing needed is clearly on the order of the width of the variable region, which is itself linear in σ.) For any adjacent pair of transmission lines of the same parity (that is, both come from positive occurrences or both from negative occurrences in their clauses), we can coalesce the transmission lines into one with the following merge unit:



If either of the incoming transmission lines comes from a selected literal, the outgoing transmission line will continue transmitting the information. Now we can simplify the incoming transmission lines so that they alternate positive and negative literals.

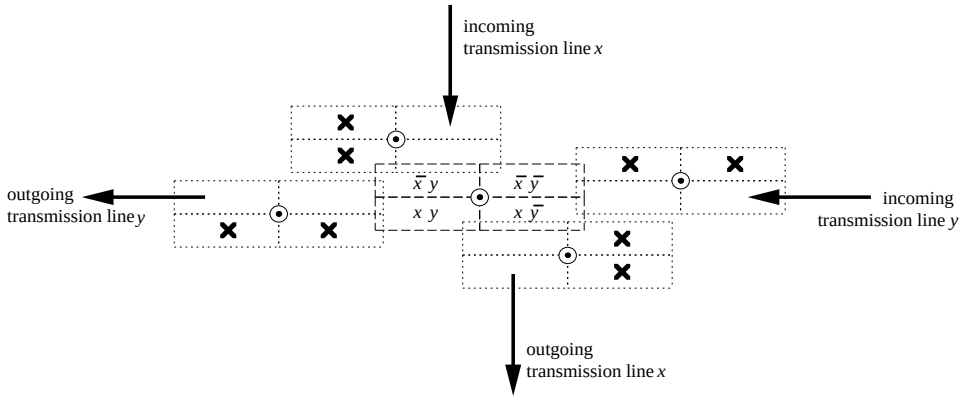


Furthermore, we can coalesce all of the positive literal transmission lines into a single line.

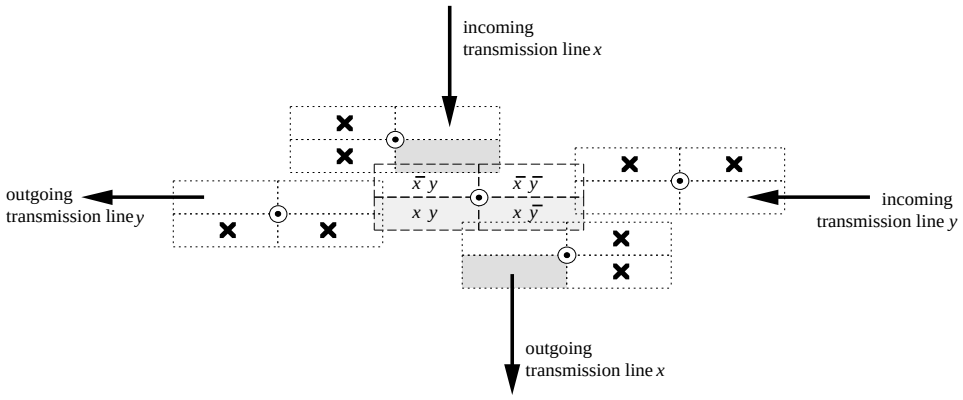


The next step is to coalesce the negative literal transmission lines. But this requires us to cross transmission lines. For this purpose we use the following crossover unit:[4]

[4] We are indebted to Stanley Chen for a dramatic simplification and generalization of our original crossover, which involved 15 labeled and 22 unlabeled points, to the current one of 5 labeled and 8 unlabeled points.



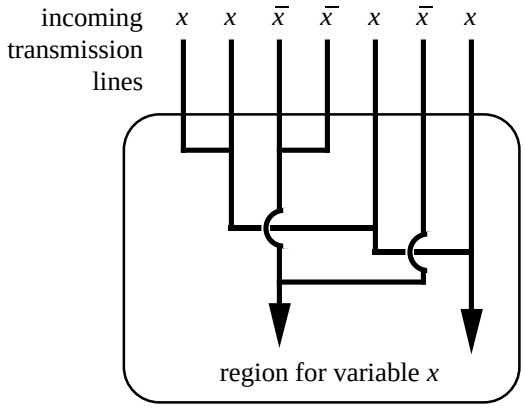
This unit has the following behavior. If the x incoming transmission line is selected, then the central point must be labeled in one of the two positions marked $x\ y$ and $x\ \overline{y}$. In either case, the lower right point must be labeled in the outgoing transmission line's selected state, as shown below:



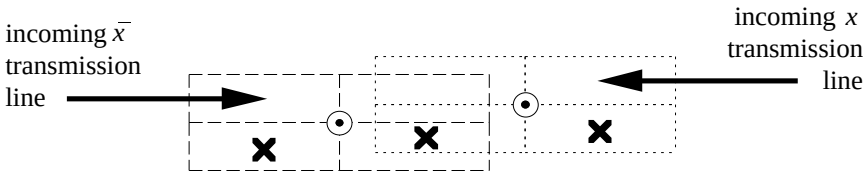
Note that the outgoing transmission line for x is thereby selected. Similarly, selection of the y transmission line is passed through from upper right to lower left.

Using the crossover unit, we can extend the variable region layout as follows:[5]

[5] Some comment is warranted about the planarity or nonplanarity of the layout. There are two graphs being used in the construction, the bipartite graph of clauses and variables, and a more articulated graph of transmission lines. Although the bipartite graph is planar, the use of crossovers induces nonplanarity within a subgraph of the more articulated graph, namely, the subgraph corresponding to a single node of the bipartite graph (a variable



Now, we have successfully coalesced all the x and $\overline{x}$ transmission lines into a single line for each. We must only guarantee that both lines are not selected, which we can do by connecting the ends of the two lines with a junction unit:



If both lines are selected, this unit cannot be labeled.

Each variable region requires at most $O(\sigma)$ merge units and crossover units and one junction unit. The extra transmission line routing is $O(\sigma)$ for each transmission line, giving a total of $O(\sigma^2)$ points per variable region. The total number of points needed for all variable regions, then, is $O(\sigma^2|U|)$.

This completes the construction of a label-placement problem from a planar 3-SAT problem. The constructed point set can be admissibly labeled just in case we can select one literal point from each clause unit and label it in the shaded region. Consider a potential choice of selected literals. If they are inconsistent (that is, some x literal and some $\overline{x}$ literal are selected) then by virtue of the merging the single final x transmission line will be selected as will the single final $\overline{x}$ line. Hence, the junction unit will be unlabelable. In

region). The transmission line graph as a whole is thus nonplanar, but we can still rely on the complexity results of planar graph layout, as the nonplanarity is localized within nodes of the planar bipartite graph.

summary, if the choice of literals is inconsistent, the system is unlabelable. Thus, the system is labelable just in case the original formula is satisfiable.

The total number of points needed is $O(|C|)$ for the clause units, $O(\sigma^3)$ for the transmission lines, and $O(\sigma^2|U|)$ for the variable regions, the sum of which is $O(\sigma^3)$, that is, $O((|U| + |C|)^3) < O((|U| \cdot |C|)^3)$. Thus, the construction of the label-placement problem for a given planar 3-SAT problem is polynomial in the size of the 3-SAT problem. Thus, if label placement were solvable in polynomial time, so would planar 3-SAT. Label placement is thus an NP-hard problem.

This concludes the argument that SLP label placement is NP-hard, and thus NP-complete. The NP-completeness of Zoraster's problem and ALP are immediate corollaries.

5 Complexity of Optimal Label Placement

We can use the NP-completeness of SLP to provide bounds on the complexity of the optimization version of the label placement problem, OLP.

A lower bound is straightforward; clearly, any algorithm for OLP solves SLP as well. Thus, OLP cannot be solved in polynomial time unless $P = NP$. As it turns out, the other direction holds as well. That is, if $P = NP$ then OLP can be solved in polynomial time; OLP is "NP-easy" [4]. In summary, although OLP is not an NP-complete decision problem, it has the hallmark of one, solvability in polynomial time if and only if $P = NP$.

We prove this by considering an intermediate decision-problem variant of OLP, the problem of determining if there is a point-feature labeling of cost $B \in \mathbb{Z}^+$ or less. We state a general version of this decision problem

below that also incorporates the notion of extending a partial labeling:

> **Extendible Optimal Point-Feature-Label Placement** (EOLP)
> *Instance:* A set P of point features each with a position in $\mathbb{R}^2$ and a label size, each of which can be labeled in one of a set N of relative positions (for example, the eight relative positions in Figure 2); cost functions c_{pl} and c_{ll} satisfying (1) and (2), respectively; a partial labeling $\Lambda_k = \{(p_{\lambda_1}, n_{\lambda_1}), (p_{\lambda_2}, n_{\lambda_2}), \ldots, (p_{\lambda_k}, n_{\lambda_k})\}$; and a cost limit, B.
> *Question:* Is there a label placement $g : P \rightarrow N$ that is a superset of Λ_k and that has cost
>
> $$\sum_{p \in P} c_{pl}(p, g(p)) + \sum_{p,q \in P} c_{ll}(p, g(p), q, g(q)) \leq B?$$

It is easy to show that EOLP is in NP and that an arbitrary instance of ALP can be reduced in polynomial time to an instance of EOLP, thereby establishing the NP-completeness of EOLP. These straightforward results are left for the reader to prove. We use the NP-completeness of EOLP to show that OLP is NP-easy.

Assume that we have a procedure S_{EOLP} for solving EOLP. Let C_{max} be the largest integer that appears in the definition of c_{pl} and c_{ll} in the statement of an arbitrary instance of OLP. Each of the $|P|$ labels may be responsible for as many as $|P| + 1$ non-zero terms in the cost formula, each with maximum value C_{max}. We can therefore state the following bounds on the optimal labeling cost, B_{opt}: $0 \leq B_{opt} \leq C_{max}(|P|^2 + |P|)$. We can determine the cost of an optimal labeling for an arbitrary instance of OLP by setting up a corresponding instance of EOLP (with $\Lambda = \emptyset$) and using procedure S_{EOLP} and binary search. This approach requires at most $\lceil \log_2(C_{max}(|P|^2 + |P|)) \rceil$ calls to S_{EOLP}, a number that is polynomial in the length of the problem statement.[6]

Having used S_{EOLP} to compute B_{opt} for an arbitrary instance of OLP, we can go on to use corresponding instances of EOLP and procedure S_{EOLP} to determine an optimal labeling. A partial labeling that is a subset of an optimal labeling is called an *extendible partial labeling*. Given an extendible partial labeling $\Lambda_k = \{(p_{\lambda_1}, n_{\lambda_1}), (p_{\lambda_2}, n_{\lambda_2}), \ldots, (p_{\lambda_k}, n_{\lambda_k})\}$, we can determine a labeling for point-feature $p_{\lambda_{k+1}}$ that results in a new extendible partial labeling $\Lambda_{k+1} = \{(p_{\lambda_1}, n_{\lambda_1}), (p_{\lambda_2}, n_{\lambda_2}), \ldots, (p_{\lambda_k}, n_{\lambda_k}), (p_{\lambda_{k+1}}, n_{\lambda_{k+1}})\}$

[6] A suitable definition for the length of an instance of OLP is $|P| + \lceil \log_2(C_{max}) \rceil$.

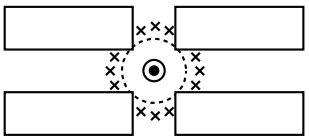
as follows: by labeling $p_{\lambda_{k+1}}$ in each of the $|N|$ possible ways, we can generate $|N|$ candidates for Λ_{k+1}; using at most $|N|-1$ corresponding instances of EOLP and $|N|-1$ calls to S_{EOLP} we can then determine which of these candidates is in fact an extendible partial labeling. (At least one of them must be, and it does not matter which one we choose.) This procedure can be started with $\Lambda_0 = \emptyset$, and will terminate with an optimal labeling $\Lambda_{|P|}$ having cost B_{opt}, using at most $|P|(|N|-1)$ calls to S_{EOLP}.

Thus an instance of OLP can be reduced to a polynomial number of instances of EOLP, a problem we already know to be NP-complete. OLP is therefore NP-easy. Given that OLP is also NP-hard, we see that OLP can be solved in polynomial time if and only if $P = NP$.

6 Generalizing the NP-Completeness Proof

Although we have shown that SLP, ALP, and OLP are (probably) intractable, there may be other instances of OLP that are more tractable. In this section, we demonstrate that certain other natural instances are at least as hard. First, the NP-hardness proof for SLP relied on using a set of unlabeled point features. If we restrict the problem further to disallow unlabeled point features (so that the set U is empty), the problem remains NP-hard. The appendix provides the appropriate constructions for clause units, transmission lines, crossover units, and so forth.

Some presentations of label placement algorithms use more label positions, say by adding positions directly to the east and west. We can always reduce such problems to SLP by adding unlabeled points to disallow use of the extra positions. This works even for certain methods that assume an unbounded number of positions, say all positions tangent to a unit circle around the point (as proposed by Hirsch [5]). We can add a finite number of unlabeled points to bound the usable label positions to within an arbitrarily small variance from the standard four positions, as shown:[7]



[7] Note that infinite precision is not needed in specifying the positions of the unlabeled points in this construction, because we do not have to restrict the labels to exactly the four standard label positions. Variation from the standard positions by a sufficiently small error still allows the several constructions to work.

Finally, we may hope that there is an efficient algorithm that, though it cannot guarantee an optimal solution to OLP, can generate one within some factor ε. This is also not possible unless $P = NP$. We go into this proof in more detail.

An *ε-approximation algorithm* for an optimization problem A is a polynomial-time algorithm with the property that

$$\frac{C_{approx}(A) - C_{opt}(A)}{C_{opt}(A)} <= \varepsilon,$$

where $C_{approx}(A)$ is the cost found by the approximation algorithm, and $C_{opt}(A)$ is the optimal cost [11]. In this section we prove that unless $P = NP$, there is no ε-approximation algorithm for OLP.

The proof is by contradiction. Assume there is an ε-approximation algorithm for OLP. We show below that such an algorithm can be used to solve SLP in polynomial time.

We do this by taking an instance of SLP and converting it to a similar instance of OLP. Informally, we construct an instance of OLP in which each potential overlap (as enumerated in the corresponding instance of SLP) is assigned a cost of more than $\lceil \varepsilon |P| \rceil$, where $|P|$ is the number of point features to be labeled. More formally, let an instance of SLP be defined by sets P and U and numbers X and Y, which in turn define functions c_{pl} and c_{ll}, and let the corresponding instance of OLP be defined by the same set P, the set N being the four positions used in SLP, and functions c'_{pl} and c'_{ll} defined as follows:

$$c'_{pl}(p, n) = \begin{cases} 1 & \text{if } c_{pl} = 0 \\ 2 + \lceil \varepsilon |P| \rceil & \text{if } c_{pl} = 1 \end{cases}$$

$$c'_{ll}(p, n, p', n') = \begin{cases} 0 & \text{if } c_{ll} = 0 \\ 1 + \lceil \varepsilon |P| \rceil & \text{if } c_{ll} = 1 \end{cases}$$

We can then use the approximation algorithm to solve instances of OLP created by this conversion process. Suppose the algorithm finds a labeling of cost $|P|$. Then we know there is an admissible labeling of all the point features, i.e., a solution to the corresponding SLP problem.

Suppose the algorithm returns a labeling of cost greater than $|P|$. This could be because of a point-label overlap or a label-label overlap. In the former case, the cost of this labeling has to be at least $|P| - 1 + (2 + \lceil \varepsilon |P| \rceil) > 1 + (1 + \varepsilon)|P|$. In the latter case, the cost of the labeling has to be at least

$|P| - 0 + (1 + \lceil \varepsilon |P| \rceil) > 1 + (1+\varepsilon)|P|$ once again. Now if there is an admissible labeling, the cost of the corresponding optimal labeling will be $|P|$. Thus if there is an admissible labeling the approximation algorithm cannot return a labeling of cost greater than $|P|$, because the labeling returned by the approximation algorithm would then not meet its own defining criterion, because

$$\frac{(1 + (1+\varepsilon)|P|) - |P|}{|P|} = \frac{1}{|P|} + \varepsilon > \varepsilon.$$

So if the algorithm returns a labeling of cost greater than $|P|$, there cannot be an admissible point labeling.

Thus the polynomial-time approximation algorithm for OLP can be used as a polynomial-time decision procedure for SLP. But this is a contradiction, because we know the latter problem is NP-complete. Therefore there can be no ε-approximation algorithms for OLP—regardless of the size of ε—unless $P = NP$. Whether an efficient approximation algorithm exists when the cost functions are restricted in some way (e.g., when each overlap has unit cost) is an open problem.

7 Conclusion

We have presented a proof that a particularly simple variant of the cartographic label placement problem is NP-complete, hence, the more general versions, including determining admissible or optimal label placements are computationally intractable. Several generalizations of the problem, including determining approximately optimal placements, or placements under other simplifications of the problem, also reduce to NP-hard problems. Thus, unless $P = NP$, none of these problems has a polynomial algorithm for its solution, but if $P = NP$ all do. To solve label-placement problems, we must therefore look either to heuristic methods or to further constraints on the problem (perhaps arising from the structure of problems that appear in actual maps) that allow for efficient computation of solutions.

References

[1] J. Ahn and H. Freeman. A program for automatic name placement. *Cartographica*, 21(2&3):101–109, Summer&Autumn 1984. Originally published in *Proceedings of the Sixth International Symposium on Automated Cartography (Auto-Carto Six)*, Ottawa/Hull, October 1983.

[2] Laurence W. Carstensen. A comparison of simple mathematical approaches to the placement of spot symbols. *Cartographica*, 24(3):46–63, 1987.

[3] B. Chazelle. The polygon containment problem. In F. P. Preparata, editor, *Advances in Computing Research, Vol. 1: Computational Geometry*, pages 1–33. JAI Press, Greenwich, Connecticut, 1983.

[4] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, New York, 1979.

[5] Stephen A. Hirsch. An algorithm for automatic name placement around point data. *The American Cartographer*, 9(1):5–17, 1982.

[6] Eduard Imhof. Die Anordnung der Namen in der Karte. *International Yearbook of Cartography*, 2:93–129, 1962.

[7] Eduard Imhof. Positioning names on maps. *The American Cartographer*, 2(2):128–144, 1975.

[8] C. Jones. Cartographic name placement with Prolog. *IEEE Computer Graphics and Applications*, 9(5):36–47, September 1989.

[9] Nils J. Nilsson. *Problem-Solving Methods in Artificial Intelligence*. McGraw Hill, New York, 1971.

[10] Christos H. Papadimitriou. The Euclidean TSP is NP-complete. *Theoretical Computer Science*, 4:237–244, 1977.

[11] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, Englewood Cliffs, New Jersey, 1982.

[12] Diane L. Souvaine and Christopher J. Van Wyk. How hard can it be to draw a pie chart? *Mathematics Magazine*, 63(3):165–172, June 1990.

[13] Roberto Tamassia and Ioannis G. Tollis. Planar grid embedding in linear time. *IEEE Transactions on Circuits and Systems*, 36(9):1230–1234, September 1989.

[14] Jan W. van Roessel. An algorithm for locating candidate labeling boxes within a polygon. *The American Cartographer*, 16(3):201–209, 1989.

[15] Pinhas Yoeli. The logic of automated map lettering. *The Cartographic Journal*, 9(2):99–108, December 1972.

[16] Steven Zoraster. Integer programming applied to the map label placement problem. *Cartographica*, 23(3):16–27, 1986.

A Proof Constructions Without Unlabeled Points

The proof can be made more general by eliminating its reliance on unlabeled points. This problem variant is given as follows:

Simpler Admissible Point-Feature-Label Placement
Instance: A set P of point features each with a position in $\mathbb{R}^2$, and each of which can be labeled in one of the 4 relative positions given in Figure 2(a); numbers X and Y that give the fixed horizontal and vertical extent of a label.
Question: Is there a label placement $g : P \to N$ such that

$$\sum_{p \in P} c_{pl}(p, g(p)) + \sum_{p,q \in P} c_{ll}(p, g(p), q, g(q)) = 0$$

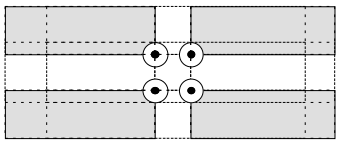
where

$$c_{pl}(p, n) = \begin{cases} 1 & \text{if there is a point } p' \in P \\ & \quad \text{such that } p' \in region(p, n) \\ 0 & \text{otherwise} \end{cases}$$

and

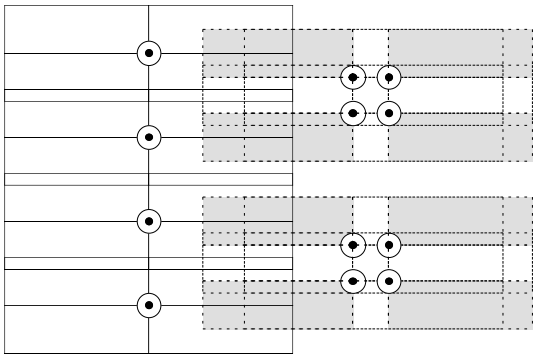
$$c_{ll}(p, n, p', n') = \begin{cases} 1 & \text{if } region(p, n) \cap region(p', n') \neq \emptyset \\ 0 & \text{otherwise} \quad ? \end{cases}$$

To eliminate the reliance on unlabeled points, we modify all of the various constructions to remove unlabeled points. A useful new construction is a *stop unit*, a simple layout of four labeled points that has a completely forced labeling.



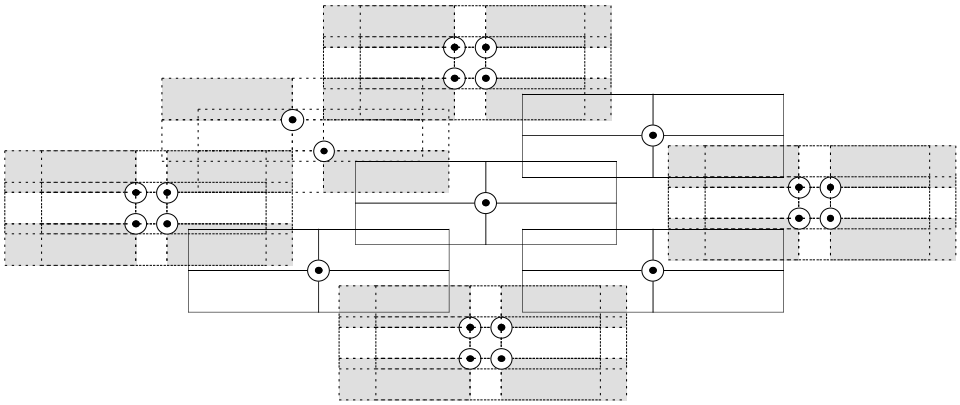
Stop units can be used to replace unlabeled points as the shaded label positions (which are forced) can overlay any label regions we wish to render invalid, as long as there is room for the extra points and labels of the stop unit. The constructions in the text were designed so as to put all unlabeled points near the periphery of the constructions so that room is available.

For the transmission line construction, for instance, we can align a series of stop units on one edge of the line.



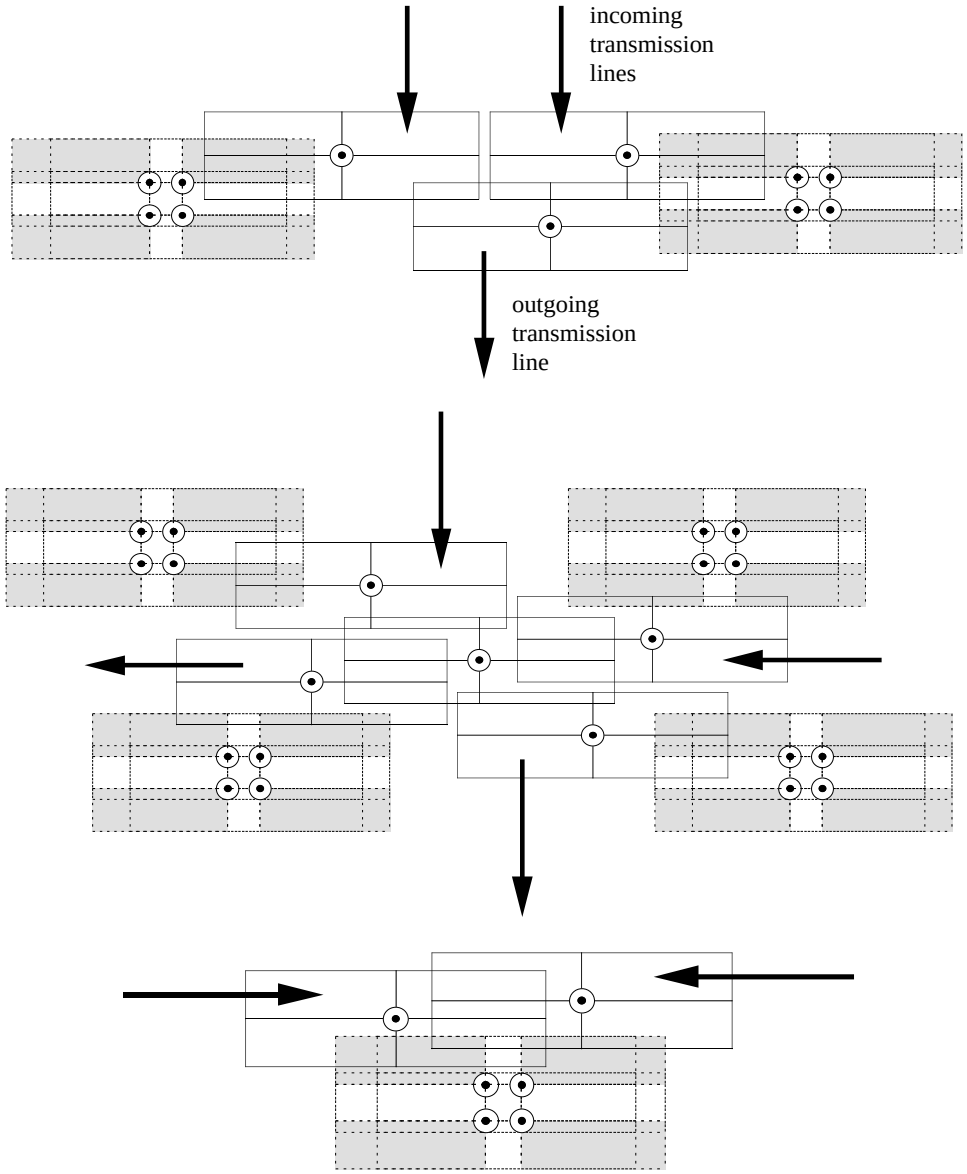
(The ability to swap sides in a transmission line, or turn corners, is obviously still possible with the new construction.)

The clause unit is augmented with four full stop units and a "partial" stop unit (two of the points being made superfluous by neighboring stop units) that serve the function of the original seven unlabeled points.



There is still room for transmission lines from the three outside label regions.

The merge, crossover, and junction units are especially simple to construct using stop units instead of unlabeled points.



The increased sizes of the constructions only introduces a constant into the reduction. Hence, the proof is preserved using these constructions, and we can dispense with unlabeled points.