On the Edge Label Placement Problem [*]

Konstantinos G. Kakoulis and Ioannis G. Tollis

Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75083-0688
email: `kostant@utdallas.edu`, `tollis@utdallas.edu`

Abstract. Let $G(V, E)$ be a graph, and let $f : G \rightarrow R^2$ be a *one to one* function that produces a layout of a graph G on the plane. We consider the problem of assigning text labels to every edge of the graph such that the quality of the labeling assignment is optimal. This problem has been first encountered in automated cartography and has been referred to as the Line Feature Label Placement (LFLP) problem. Even though much effort has been devoted over the last 15 years in the area of automated drawing of maps, the Edge Label Placement (ELP) problem has received little attention. In this paper we investigate computational complexity issues of the ELP problem, which have been open up to the present time. Specifically we prove that the ELP problem is NP-Hard.

1 Introduction

In recent years graph drawing has received increasing attention due to the large number of applications, such as, entity relationship diagrams, software engineering diagrams, CASE tools, debugging tools, communication networks, and database design [3]. The labeling of the graphical features of a drawing in most of these applications is essential, since it gives important information about the relations represented by the drawing.

The problem of labeling graphs can be divided into two subproblems:

- **NLP**: (Node Label Placement) is the problem of placing text labels assigned to particular nodes of the graph.
- **ELP**: (Edge Label Placement) is the problem of placing text labels assigned to particular edges of a graph.

Most of the research addressing the above labeling problem has been done on labeling features of geographical and technical maps. Christensen, Marks and Shieber present a comprehensive survey of algorithms for the labeling problem [2].

The NLP problem has been the subject of extensive research in recent years, and the complexity issues of that problem have been well documented. It has

[*] Research supported in part by NIST, Advanced Technology Program grand number 70NANB5H1162.

been proven independently by three different groups [6, 11, 13] that the NLP problem is NP-Complete even in its simplest form.

Even though much effort has been placed in solving the NLP problem, the ELP problem has received little attention [14, 16]. Many heuristics devised to solve the ELP problem [1, 4, 5, 7, 8], are based on exhaustive search algorithms with backtracking. These algorithms do not produce the desirable results, due to the tendency of those methods to get trapped in local optima. They also take exponential time.

Definitely in geographical maps, the labeling of lines (ELP problem) is less restrictive than the labeling of points (NLP problem) due to the fact that lines are usually very long and thus there are many alternative positions to place labels. But the edge density of a drawing can vary, and edge labeling algorithms must perform well even for a high edge density drawing, since unlike in geographical maps the size and density of the edges (lines) of a drawing can and usually are restrictive with respect to label positioning. This makes most of the algorithms in automated cartography for the ELP problem inefficient. Another approach is to adopt techniques used to solve the NLP problem [2] in order to solve the ELP problem.

Up to the present time, the complexity of ELP is an open problem. The NLP and ELP problems have very similar structure. However, as we and other members of the community [12] observed, a direct transformation from NLP to ELP seems very difficult. In this paper we prove that the ELP problem is NP-Hard.

2 The ELP (`Edge Labeling Placement`) problem

2.1 Labeling quality

The ELP problem is the problem of assigning text labels to any edge of a predefined layout of a graph (the equivalent problem for geographical and technical maps is the LFLP problem) such that the association of the labels to their corresponding edges is clear.

It is very important to note that the visual inspection of a labeling assignment must be sufficient to explain the semantics of any label. Let us consider a labeling assignment of a set of edges where each edge is a street in a city and each label is the name of that street. Then a visual inspection of the labeling assignment must unambiguously reveal the name of each street.

There has been extensive effort especially by cartographers like Imhof [10] and Yoeli [15], to devise rules that measure the semantic clarity of a labeling assignment. Three concepts stand out, and are the basic rules that in general give an accurate assessment of the semantics of labels.

Basic rules for labeling quality [10, 15]:

1. No overlaps of a label with other labels or other graphical features of the layout are allowed.

2. Each label can be easily identified with exactly one graphical feature of the layout (i.e., the assignment is unambiguous).
3. There is a ranking of all potential labels for any particular edge.

A label respects the first rule if it does not overlap any graphical feature, even though it is allowed to touch the edge that it belongs to. Also, a label respects the second rule if it is placed very close or touches the edge that it belongs to.

The ranking of the potential labels for an edge typically captures the aesthetic preference of those labels, which is an essential criterion for the labeling quality of geographical and to some extend technical maps. It also allows to introduce problem specific constraints (i.e., the label of an edge must be closer to the source or destination node). The ranking of a label depends only on its position with respect to its associated edge and is not influenced by overlaps. We associate a penalty for each label according to its ranking.

Since the layout of a graph is fixed, there are instances where even an optimal assignment produces labels that do not strictly follow those rules. In that case we want to have a way of evaluating how good is a given labeling assignment. We can do that by assigning to each label a quantity that evaluates the quality of that label. We name this quantity $COST$. $COST(i, j)$ is a function that gives us the penalty of assigning label j to edge i in the final labeling assignment. The $COST$ for each label j is a linear combination of:

- The penalty with respect to the ranking of label j.
- The penalty which reflects the severity of the violation of the first two basic rules for label j.

Now, let $rank(i, j)$ be the ranking of label j among all potential labels of edge i, where $0 \leq rank(i, j) \leq C$, $C \in \mathbb{R}$. Then $a * rank(i, j)$ is the penalty with respect to the ranking of label j, where a is a constant. Also, let $b * clarity(i, j)$ be the penalty with respect to the second basic rule, if label j is assigned to edge i, where b is a constant. Finally, let $pen(i, j, k, l)$ be the penalty if label[2] j of edge i overlaps label l of edge k. Then:

$$COST(i,j) = a\, rank(i,j) + b\, clarity(i,j) + \sum_{k \in E, k \neq i} \sum_{j \cap l \neq \emptyset} pen(i,j,k,l)\, P(k,l)$$

Where:

$$P(k,l) = \begin{cases} 1, & if\ label\ l\ is\ assigned\ to\ edge\ k \\ 0, & otherwise \end{cases}$$

The last condition guarantees that only labels assigned to a final labeling assignment affect the cost with respect to the labeling quality.

[2] We consider as potential labels only labels that do not overlap any other graphical feature, except other labels and(or) their corresponding edge.

2.2 Label positioning

In order to completely describe the labeling problem we need to define how we construct the set of potential labels for any edge. In *Fig.* 1(*a*) labels *A*, *B*, *C*,

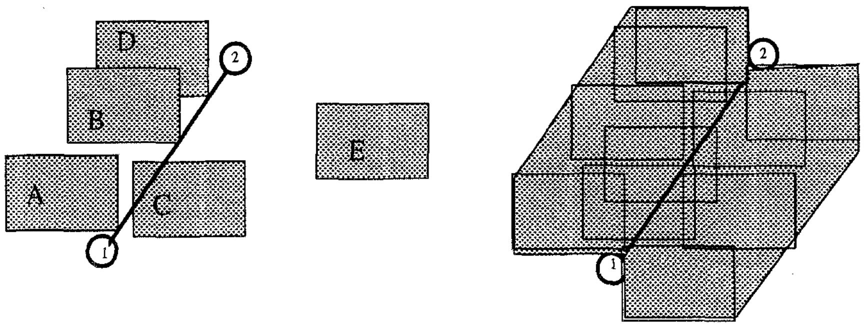

Fig. 1. (a) How to position labels for an edge; (b) Labeling space of an edge.

D and *E* are potential labels for the edge (1,2). Labels *A*, *B* and *D* follow the first two rules. Instead, label *C* intersects the edge that belongs to, which is a violation of the first rule. If a drawing has more than one edges, and labels like *E* floating between edges , it is difficult to have a clear understanding to which edge that label belongs to. So labels like *E* violate the second rule, and must be avoided.

We can define the set of all potential labels for a given edge in the *discrete* or *continuous* labeling space. In the *discrete* labeling space the set of all potential labels is finite and each label is identified by its position on the layout, like the labels in *Fig.* 1(*a*). In the *continuous* labeling space the set of all potential labels is infinite, and for each edge we define a region which is bounded by a closed line, where each potential label for that edge must lay inside that region. Candidate labels are labels that have at least one intersection point with their associated edge. By imposing this restriction we avoid labels like *E* in *Fig.* 1(*a*). The shaded region around the edge (1,2) in *Fig.* 1(*b*) is the *continuous* labeling space for that edge, and any label that is placed inside that shaded region is a candidate label for that edge.

2.3 Formulation of the ELP problem

The ELP problem is an optimization problem since the objective is a labeling assignment of minimum cost. Each label which is part of a final assignment carries a penalty that the *COST* function calculates. The objective is to find a set of labels, one for each edge, that yields minimum total cost. We first consider the problem that optimizes the cost of a labeling assignment.

The Optimal ELP Problem : Let $G(V, E)$ be a graph and let $f : G \rightarrow R^2$ be a *one to one* function that produces a drawing for graph G. Let W_i be the set of all potential labels for any edge i.
Question : Find a labeling assignment that minimizes the following function:

$$\sum_{i=1}^{|E|}\sum_{j=1}^{|W_i|} COST(i,j)P(i,j)$$

Where:

$$P(i,j) = \begin{cases} 1, & if\ label\ j\ is\ assigned\ to\ edge\ i \\ 0, & otherwise \end{cases}$$

and

$$\sum_{i=1}^{|E|}\sum_{j=1}^{|W_i|} P(i,j) = |E|$$

and

$$\sum_{j=1}^{|W|} P(i,j) = 1, \qquad 1 \leq i \leq |E|.$$

The last two conditions guarantee that any edge will have exactly one label assigned to it, since $P(i,j) = 1$ if and only if label j is assigned to edge i. The ELP problem as stated is combinatorial in nature, even though the underlying geometry gives meaning to the cost function, the interpretation of the cost function can be regarded as independent from some particular geometry. In our case we will always interpret the cost function with respect to Euclidean geometry.

Now we will impose some extra constraints in order to obtain a simpler version of the ELP problem. Here we are interested to find if there is an admissible labeling assignment where each label is of zero cost, with respect to the first two basic rules for labeling quality, rather than finding an assignment of optimal cost. This simplification transforms the ELP problem into a decision problem, which enables us to investigate the computational complexity aspects of the problem.

First we redefine the $COST$ function as follows:

$$COST(i,j) = \begin{cases} 0, & if\ rule\ 1\ and\ 2\ are\ followed \\ 1, & otherwise \end{cases}$$

The resulting ELP problem becomes the *Admissible* ELP (AELP) problem. Here the objective is to find a labeling assignment of zero cost with respect to the labeling quality.
The Admissible ELP Problem :
Question : Find a labeling assignment such that:

$$\sum_{i=1}^{|E|}\sum_{j=1}^{|W|} COST(i,j)P(i,j) = 0,$$

subject to the same constraints as above.

Next we further restrict the AELP problem by:

- requiring the labels of each edge to be of the same size.
- restricting the size of the set of potential labels for each edge by not allowing potential labels of the same edge to overlap (i.e., in *Fig.* 1 labels B and D are not considered both as potential labels of edge (1,2)).

The latter constraint guarantees that each edge has a discrete number of potential labels. Hence, the resulting AELP problem becomes the *Discrete* AELP (DAELP) problem.
The Discrete AELP (DAELP) problem : Let $G(V,E)$ be a graph and let: $f: G \longrightarrow R^2$ be a *one to one* function that produces a drawing for graph G. Let W_i be the set of all potential labels for any edge i.
Question : Find a labeling assignment such that:

$$\sum_{i=1}^{|E|}\sum_{j=1}^{|W_i|} COST(i,j)P(i,j) = 0$$

Where:

$$P(i,j) = \begin{cases} 1, & if\ label\ j\ is\ assigned\ to\ edge\ i \\ 0, & otherwise \end{cases}$$

and

$$\sum_{i=1}^{|E|}\sum_{j=1}^{|W_i|} P(i,j) = |E|$$

and

$$\sum_{j=1}^{|W|} P(i,j) = 1, \qquad 1 \leq i \leq |E|.$$

In the next section we prove that the DAELP problem is NP-Complete.

3 The NP-Completeness of the DAELP problem

We will prove that the DAELP problem is NP-Complete by transforming the 3-SAT problem [9], a well known NP-Complete problem, to it. Recall that 3-SAT is defined as follows:
Instance: Set X of variables, collection U of clauses over X, such that each clause has exactly 3 literals.
Question: Is there a satisfying truth assignment for U?

In order to transform 3-SAT into DAELP problem, we do the following: For each variable in an instance of the 3-SAT problem we transmit to any clause that contains a literal of this variable the information on the status of this variable. This transmition takes place through the construction of a transmition network. The goal is to associate the satisfiability of the 3-SAT problem with the existence of an edge labeling assignment of zero cost for the transmition network. Generally speaking each variable will be linked with all clauses that contain its complement through a route, such that once each variable has been assigned a value, there is only one possible labeling assignment of zero cost for each route. By knowing how each route has been labeled we can conclude the satisfiability of any clause that this route is connected by observing if there is enough room for that clause to have a label of zero cost assigned to it.

To construct the transmition network we need some basic building blocks. The interconnection of these blocks will produce the final transmition network.

The first building block is the **variable_block** shown in *Fig.* 2. Each variable and its complement will be represented by a *variable_block*. Let a *variable_block* represent X and $\overline{X}$. If $X = TRUE$ then edge $(1, 2)$ is assigned label *head*. If $\overline{X} = TRUE$ then edge $(1, 2)$ is assigned label *tail*. Every edge in a *variable_block*

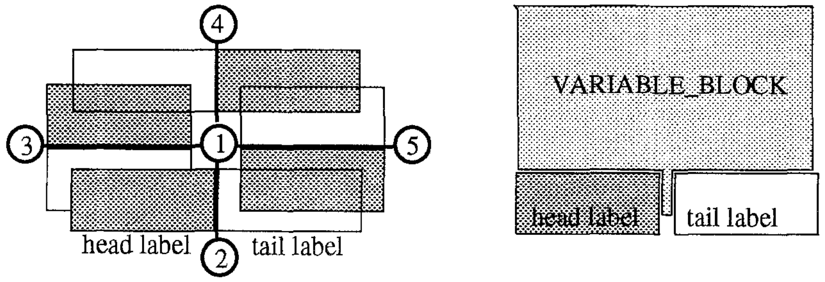

Fig. 2. An example of a variable_block.

has 2 labels of potential zero cost. Because of the cyclic structure and the way labels overlap there can be only two solutions to the DAELP problem for any *variable_block*. One solution contains all the shaded labels and the other all the non-shaded labels, since any shaded label is overlapped by a non-shaded label. Due to space limitations, we present the following results without proofs.

Lemma 1. *For a variable_block we have:*

1. *If a variable has been assigned a value, then the variable_block that represents that variable has only one labeling assignment of zero cost.*
2. *In a labeling assignment of zero cost only one of the head or tail labels is part of this assignment.*

Each clause will be represented by a **clause_block** as shown in *Fig.* 3. Edge $(1, 2)$ has 3 potential labels (X,Y,Z) of zero cost, one for each literal in that clause. If a label of zero cost is assigned to edge $(1, 2)$ then the clause represented by *clause_block* is satisfied.

The following building blocks will serve as channels that transfer the truth assignment of the variables (*variable_blocks*) to the clauses (*clause_blocks*).

First we introduce the **pipe_block** shown in *Fig.* 4. Each edge has only 2 potential labels of zero cost. Also each label overlaps with exactly one other label.

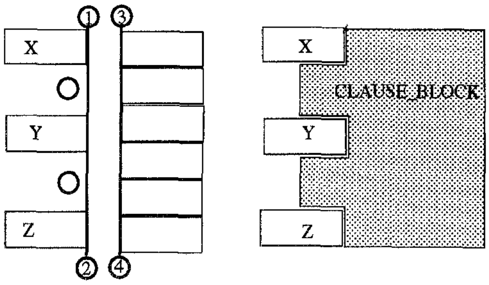

Fig. 3. An example of a clause_block.

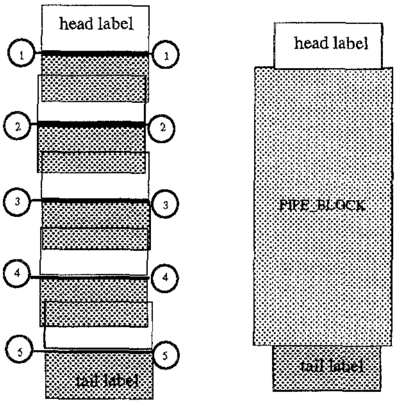

Fig. 4. An example of a pipe_block.

For the *pipe_block* we have a numbering of the edges that reveals the following pattern: Except for the first and last edge in the sequence, the shaded label of an edge overlaps the non-shaded label of the next edge in the sequence.

Lemma 2. *If the head (tail) label for any pipe_block is excluded from a labeling assignment of zero cost, then there exists exactly one labeling assignment of zero cost. That assignment includes the tail(head) label of that pipe_block.*

Next we introduce the **junction_block**, shown in *Fig.* 5(*a*). Notice that if one *tail* label is excluded then the *head* label must be included in a labeling assignment of zero cost for a *junction_block*. However this labeling assignment is not unique. A *junction_block* has the following properties:

Lemma 3. *For any junction_block:*

1. *If the head label is excluded then there is exactly one labeling assignment of zero cost, which includes all the tail labels of the block.*

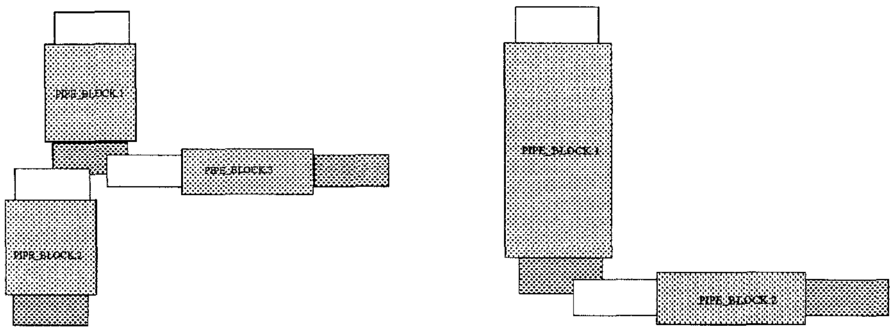

Fig. 5. (a) An example of a junction_block; (b) An example of a bend_block.

2. If all tail labels are excluded from a labeling assignment, then there is exactly one labeling assignment of zero cost that includes the head label of the block.

Next we introduce the **bend_block**, shown in *Fig.* 5(*b*), which behaves exactly the same way as a *pipe_block* does. So what holds for *pipe_blocks* certainly holds for *bend_blocks* also.

The purpose of the transmition network is to connect any variable to clauses that contain the complement of that variable. We achieve that by connecting the building blocks that we have defined so far so that they produce a bigger structure with one *head* label and more than one, if necessary, *tail* labels. The *head* label of this structure will overlap the label in the *variable_block* that represents that variable, and any *tail* label of the structure will overlap a label in some *clause_block* that corresponds to the complement of that variable in every clause that contains it.

To visualize how these connections will be made, one needs to think that for any variable we build a highway that goes from North to South and has exits only towards the east side of the highway, where all the *clause_blocks* are located. We visualize the highway as a vertical line segment and the exits as horizontal line segments. Each variable and the connections to clauses that contain its complement form a set of a highway and its exits. All *variable_blocks* stand on the same horizontal line and all the *clause_blocks* stand on the same vertical line. A highway starts from the *variable_block* and stops at the last exit. Each exit leads to a literal in a *clause_block*.

A highway with its exits is called a *serial interconnection. Figure* 6(*a*) illustrates the structure of a *serial interconnection* for variable $\overline{X1}$ that is connected to variable $X1$ in all of the clauses that contain $X1$. To build a serial interconnection one needs to:

1. replace the part of the highway where an exit occurs with a *junction_block*,
2. replace the last exit with a *bend_block*,

3. replace the rest of the highway with an appropriate number of *pipe_blocks*,
4. connect 2 consecutive blocks A1 and A2 in such a way that the *tail* label of block A1 overlaps the *head* label of A2.

By following theses rules the serial interconnection in *Fig.* 6(*b*) which is built by using the building blocks presented so far actually represents the structure in *Fig.* 6(*a*).
Note: *Every serial interconnection has one head label and as many tail labels as the number of exits.*
The *head* label of the structure is the *head* label of the first (top) building block (*Fig.* 6(*b*)). *Tail* labels are the *tail* labels for the last building blocks of any exit. The *serial interconnection* block has the following properties:

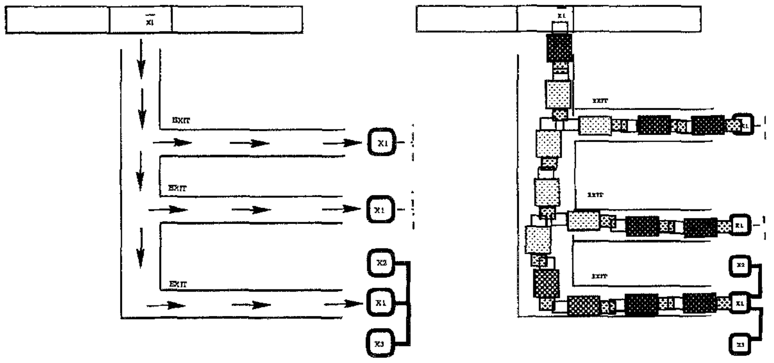

Fig. 6. (a) A serial interconnection. (b) Structure of a serial interconnection.

Lemma 4. *For a serial interconnection:*

1. *If the head label is excluded from a labeling assignment, then there is exactly one edge labeling assignment of zero cost that includes all tail labels of the structure.*
2. *If all tail labels are excluded from a labeling assignment, then there is exactly one labeling assignment of zero cost that includes the head label of the structure.*

For each literal in an instance of a 3-SAT problem there will be a serial interconnection that will represent that literal. The transmition network will be the union of these serial interconnections. *Figure* 8 shows the structure of a transmition network for an instance of a 3-SAT problem of $\{X, Y, Z\}$ variables and $\{\{\overline{X}, \overline{Y}, \overline{Z}\}, \{X, Y, Z\}, \{X, Y, \overline{Z}\}\}$ clauses.

The crossing of highways and exits is unavoidable. This is why we introduce the concept of *over − passes*. If a highway A is to the east of a highway B, and highway A has an exit a, which is south of an exit b of highway B, then exit b will intersect highway A. The building blocks defined so far are sufficient to build the network of these roads but insufficient to built over-passes. In order to be able to build such networks we need to build over-passes.

In *Fig.* 7 the *bridge_block* serves as an over-pass. The *bridge_block* possesses a very interesting property: in a labeling assignment of zero cost if we put pressure on the top (that is the *head*1 label is not available) then the pressure comes out at the bottom (that is the *tail*1 label must be included in the labeling of the *bridge_block*) regardless of which is the labeling assignment of the rest of the *bridge_block*. The same property is true if the pressure comes from any other direction. This property is essential in the construction of the transmition network, because if we replace any crossing of *serial interconnection* blocks in the transmition network with a *bridge_block* then *Lemma* 4 remains true.

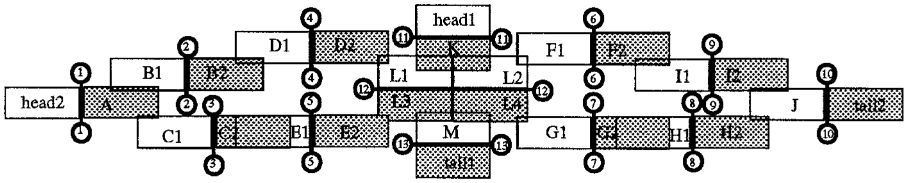

Fig. 7. An example of a bridge_block.

Lemma 5. *If the head*1*(tail*1*) or head*2*(tail*2*) label for any bridge_block is not available, then label tail*1*(head*1*) or tail*2*(head*2*) is included in any labeling assignment of zero cost.*

Now we introduce *Algorithm* 1, which given an instance of a 3-SAT problem, it constructs the transmition network for an instance of DAELP problem in polynomial time.

Algorithm 1

Given an instance of the 3-SAT problem, with n variables and m clauses:

1. For each variable we introduce a *variable_block*.
2. For each clause we introduce a *clause_block*.
3. For each *variable_block* we introduce 2 columns in the grid.
4. For each *clause_block* we introduce 3 rows (one row for each literal in the clause) in the grid.
5. For each literal we build a serial interconnection which is a collection of horizontal and vertical segments.
6. For each literal we connect the *variable_block* that corresponds to this literal with the serial interconnection for that literal in such a way that the *head* label of that serial interconnection will overlap the label in *variable_block*

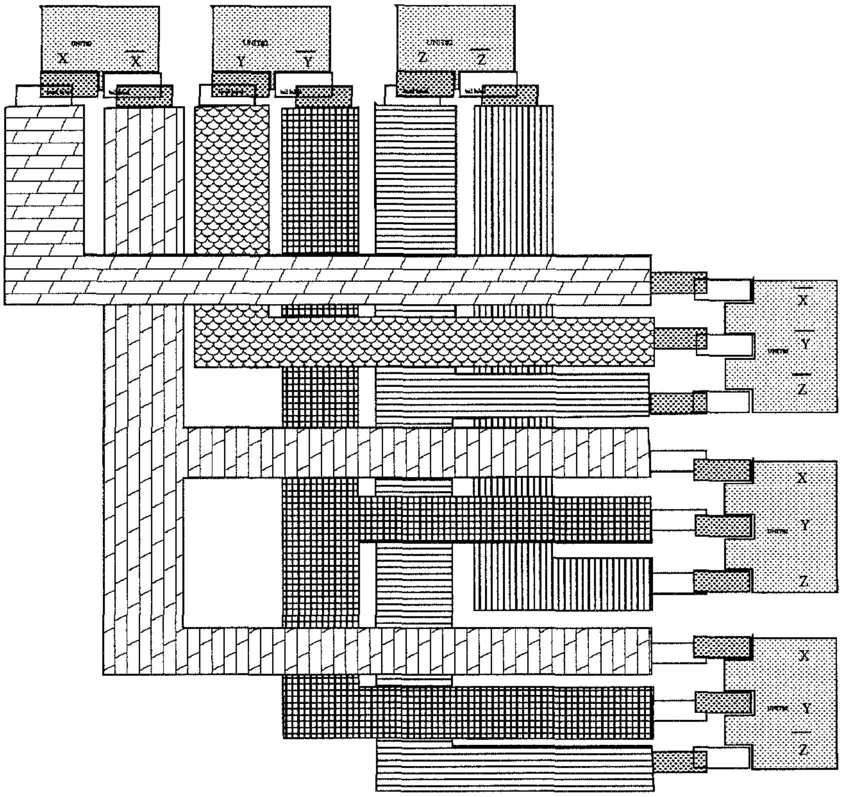

Fig. 8. A transmition network of an instance of a 3-SAT problem

that corresponds to that literal, and all labels corresponding to the complement of that literal in any clause (*clause_block*) will be overlapped by some *tail* label of that serial interconnection.

In order to build the transmition network we have to place the serial interconnection of each literal on the plane in order to be able to make the connections in *Step* 6 of *Algorithm* 1.

Step 5 of *Algorithm* 1 marks the entries of the grid that each serial interconnection occupies. Each grid entry is part of only one serial interconnection except for the entries where crossings of serial interconnections occur. That entry is assigned to a *bridge_clause*.

Step 6 of *Algorithm* 1 makes the final connections which complete the transmition network. *Figure* 8 illustrates how these connections take place: First any pair $(X, \overline{X})$ of columns in the grid is connected with the *variable_block* for variable X. The *head* label of that *variable_block* (see *Fig.* 2) overlaps the *head* label of the *serial interconnection* for X and the *tail* label of the *variable_block* overlaps the *head* label of the *serial interconnection* for $\overline{X}$. Secondly, any three rows in the grid that represent a clause are connected to a *clause_block* for that clause so each of the three labels of the clause (see *Fig.* 3) overlaps the *tail* label of some serial interconnection that ends up in that row and is associated with the complement of the literal that the label in the *clause_block* represents.

Due to space limitations we will not discuss here all the details that show that our transformation takes polynomial time. Briefly the most time consuming step of *Algorithm* 1 is *Step* 5 which can be done in $O(mn^2)$. Also, *Algorithm* 1 requires space proportional to the size of the grid. Hence we have the following:

Lemma 6. *Algorithm* 1 *produces a layout of a graph that describes an instance of 3-SAT, runs in* $O(mn^2)$ *time and requires* $O(n) \times O(m)$ *space.*

In the following theorem we prove that the ELP problem for a transmition network is equivalent to the 3-SAT problem that this network represents.

Theorem 7. *The transmition network N, constructed according to Algorithm 1, that represents an instance S of a 3-SAT problem, has an admissible edge labeling assignment of zero cost if and only if S is satisfiable.*

Proof: " $\Rightarrow$ ": By the hypothesis, N has an edge labeling assignment of zero cost. So there is a set W of labels such that $|W| = |E|$, and each edge has exactly one label assigned to it that does not overlap any other graphical feature of the drawing. Any edge in the *variable_block* has a label of zero cost assigned to it. By *Lemma* 1, either the *head* label or the *tail* label of any *variable_block* is part of the solution but not both. So for each variable represented by a *variable_block* in N we can obtain a truth assignment by examining how each *variable_block* is labeled. We claim that the truth assignment obtained this way satisfies the instance of 3-SAT associated with N.

The *head* or *tail* label of a *variable_block* is part of the solution. Any such label (which is part of the solution) overlaps the *head* label of some serial interconnection. By *Lemma* 4, that label in the *variable_block* puts pressure on that serial interconnection, and all labels in the *clause_blocks* that overlap the *tail* labels of that serial interconnection are excluded from a labeling assignment of zero cost. But any serial interconnection connects a literal (the label that overlaps the *head* label of that serial interconnection) in a *variable_block*, to the complement of that literal in any clause (labels in some *clause_blocks* that overlap some *tail* label of that serial interconnection) of that instance of the 3SAT. This implies that no label of a *clause_block* associated with a literal which has false value is part of the labeling assignment.

But by the hypothesis each edge in N has a label of zero cost assigned to it, which implies that each *clause_block* in N has a label assigned to it. Therefore, for each clause in S there exists at least one literal (the label assigned to the *clause_block* of that clause) with truth value, which implies that S is satisfiable.

" $\Leftarrow$ ": Let us assume that instance S of the 3-SAT problem is satisfiable. First we construct the transmition network N according to *Algorithm* 1. Since S is satisfiable, there is a truth assignment for every variable. By *Lemma* 1, each *variable_block* has exactly one labeling assignment of zero cost that reflects the truth assignment of S. The labeling assignment of the *variable_blocks* puts pressure (by overlapping their head labels) to those serial interconnections in N that their *tail* labels overlap the labels in any *clause_block* that correspond to literals of false value. By *Lemma* 4 this forces these serial interconnections to include all their *tail* labels to preserve a labeling assignment of zero cost. Consecutively, each label in a *clause_block* of N that corresponds to literals of false value is excluded from a labeling assignment of zero cost.

Since S is satisfiable, for each *clause_block* in N there is at least one label (corresponding to a literal which has true value) that is not overlapped by some *tail* label of the above serial interconnections.

Next, we assign all labels (that correspond to literals of true value in any clause) to their corresponding *clause_blocks*. This assignment puts pressure to all *tail* labels of any serial interconnection that its *head* label intersects a label in a *variable_block* corresponding to false value. But such labels in the *variable_blocks*, by *Lemma* 1, can't be part of a labeling assignment of zero cost. That ensures that those serial interconnections have a labeling assignment of zero cost. Finally, we remove from each *clause_block* all but one labels assigned to it.

Now each *variable_block*, each *clause_block*, and each serial interconnection has a labeling assignment of zero cost, which implies that the transmition network has a labeling assignment of zero cost since any edge in the transmition network belongs to a *variable_block* , a *clause_block* , or a serial interconnection. □

Theorem 8. *The DAELP problem is NP-Complete*

Proof: Since our transformation takes polynomial time, by *Theorem* 7 the DAELP problem is NP-Hard. Now it remains to show that the DAELP problem is in NP. One needs to guess a labeling assignment for each edge and check if it is of zero cost. The checking obviously can be accomplished in polynomial time since for each label we need to check against all the graphical features of the drawing which are of polynomial size. □

Theorem 9. *The AELP problem is NP-hard.*

Sketch of the proof: To prove the NP-Hardness of the Admissible ELP problem we follow the same steps as in the proof of the NP-Hardness of DAELP problem. We must make sure however, that all the properties of the building blocks are preserved. This can be accomplished by adding to the structure of the building blocks nodes of degree zero when edges have more than one labels on each side, as shown in *Fig*.(9). Also we need to adjust the length of each edge that has only one label at each side to be approximately equal to the height or width of the label size, as shown in *Fig*.(10). By applying these two rules we restrict the labeling space essentially down to the discrete labeling space.

As it can be observed from *Figures* 9 and 10, each edge can have an infinite number of potential labels of zero cost for the AELP problem. Also, regardless of

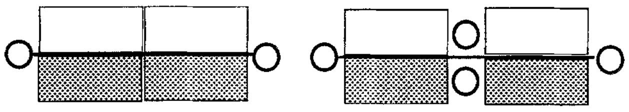

Fig. 9. An edge with more than one labels on each side.

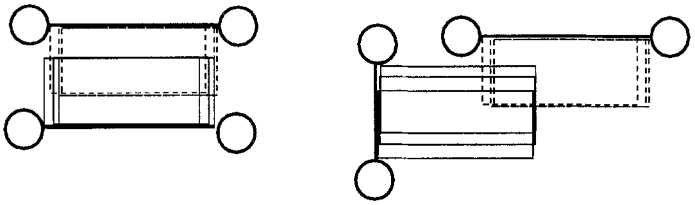

Fig. 10. Edges with one label on each side.

which label we choose for each of these edges, the behavior of these labels with respect to overlapping with other labels is exactly the same as if we had chosen labels that follow the rules of the DAELP problem. Hence, it is clear that the properties that hold for any building block for the DAELP problem, also hold for the AELP problem. □

Finally, we have the following:

Theorem 10. *The Optimal ELP problem is NP-hard.*

4 Extensions and conclusions

We have proven that a restricted version of the ELP problem, namely the AELP problem for discrete labeling space, is NP-Complete. This result implies that efficient heuristics to solve the ELP problem are needed. Even though the ELP and NLP problems are similar, which suggests that some ideas that work on NLP may work on ELP, the challenge remains to design algorithms that will take advantage of the characteristics that are specific to the ELP problem.

The research on the problem of labeling graphical features has been exclusively directed towards predefined drawings. Which is appropriate when the labeling of geographical and technical maps is the objective. But for the graph drawing community, and for anybody that draws graphs to visualize information, the labeling problem can be seen from a different perspective, since the underline geometry of any graph layout can be changed. This presents a possible dual approach. First devise graph layout techniques that reserve space for labels, and secondly devise local improvement techniques that free-up space for labels in the existing layout.

Acknowledgements

We would like to thank Joe Marks for interesting discussions.

References

1. Ahn, J. and H. Freeman, A program for automatic name placement. *Cartographica,* 21(2&3), Summer & Autumn, 1994.
2. J. Christensen, J. Marks and S. Shieber, An empirical study of algorithms for Point Feature Label Placement. *ACM Trans. on Graphics,* 14(3):203-232, July, 1995.
3. Di Battista,G., Eades P., Tamassia R., I. G. Tollis, Algorithms for Drawing Graphs: an Annotated Bibliography. *Computational Geometry, Theory and Applications,* 4(5), pp. 235-282, 1994.
4. Doerschler, J. S and H. Freeman, A rule based system for dense map name placement. *Communications of ACM,* 35(1), pp. 68-79, January, 1992.
5. Ebinger, L. R. and A. M. Goulete, Noninteractive automated names placement for the 1990 decennial census. *Cartography and Geographic Information Systems,* 17(1), pp. 69-78, January, 1990.
6. M. Forman, F. Wagner, A packing problem with applications to lettering of maps . *Proc. of 7-th Annual Symposium on Computational Geometry,* pp. 281-288, 1991.
7. Herbert Freeman, An Expert System for the Automatic Placement of Names on a Geographical Map. *Information Sciences,* 45, pp. 367-378, 1988.
8. Freeman, H. an J. Ahn, On the problem of placing names in a geographical map. *International Journal of Pattern Recognition and Artificial Intelligence,* 1(1), pp. 121-140, 1987.
9. M. R. Garey and D. S. Johnson, *Computers and Intractability*: A Guide to the Theory of NP-Completeness. W.H. Freeman and company, NY, NY, 1979.
10. Eduard Imhof, Positioning names on maps. *The American Cartographer,* 2(2), pp. 128-144, 1975.
11. Kato, T. and H. Imai, The NP-completeness of the character placement problem of 2 or 3 degrees of freedom. *Record of Joint Conference of Electrical and Electronic Engineers in Kyushu,* 1138, 1988. In Japanese.
12. J. Marks, private communication, 1996.
13. J. Marks, S. Shieber, The computational complexity of cartographic label placement. *Technical Report 05-91,* Harvard University, 1991.
14. J. W. van Roessel, An algorithm for locating candidate labeling boxes within a polygon. *The American Cartographer,* 16(3), pp. 201-209, 1989.
15. Pinhas Yoeli, The logic of automated map lettering. *The Cartographic Journal,* 9(2), pp. 99-108, December 1972.
16. S. Zoraster, The solution of large 0-1 integer programming problems encountered in automated cartography, *Operation Research,* 38(5), pp. 752-759, September-October, 1990.