# On the Multiple 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

We consider the problem of positioning text or symbol labels associated with graphical features of two dimensional maps (geographical or technical) or drawings. In many practical applications, it is often the case, that each graphical feature has more than one label. This variation of the labeling problem has not been addressed in depth. In this paper we introduce techniques that solve this problem. We have applied these techniques to drawings of graphs, and we present very encouraging experimental results.

## 1 Introduction

Traditionally, the labeling problem has been central in the framework of automated cartography. Progress has been made towards automating the label placement process in map production [1, 3, 6, 7, 10, 11, 14, 16, 18, 19, 21, 23]. Today, due to the significant growth in the area of information visualization the labeling problem may be seen through a wider scope. Labels are textual descriptions, and thus another means to convey information or clarify the meaning of complex structures. The problem of automatic label placement is important [2], and has applications in many areas including Cartography [3], GIS [13] and Graph Drawing [4].

*Research supported in part by NIST, Advanced Technology Program grant number 70NANB5H1162. A patent on these and related results is pending.

In many practical applications it is essential to associate graphical features of maps or drawings with more than one text or symbol label when these graphical features are large or long; and/or when the display of many attributes of a graphical feature is necessary. Even though many algorithms exist for the single label per graphical feature problem, very little work has been directed towards positioning more than one label per graphical feature in a map or drawing. In existing automated name placement systems for geographic maps simple techniques are being used to solve this problem [12, 14]. Specifically, each graphical feature to be labeled is partitioned into as many pieces as the number of labels for that graphical feature. Then, labeling algorithms for single label per graphical feature may be applied to the new set of partitioned graphical features. Most labeling algorithms are based on local or exhaustive search. Thus, their performance (running time and quality of solutions) is sensitive to the size of the graphical features to be labeled and to the density of the drawing. Clearly, if each graphical feature in a drawing is associated with $i$ labels, then the size of the problem is $i$ times larger, therefore, techniques based on local or exhaustive search might be slow even for small instances.

We will refer to the labeling problem where each graphical feature is associated with more than one label as the MLP (*Multiple Label Placement*) problem.

## 2 The MLP Problem.

What constitutes a good label assignment has its roots in the art of cartography. For many centuries cartographers have perfected the art of placing labels on maps. In a successful label assignment, labels must be positioned such that they are legible and follow basic aesthetic quality criteria [17, 22]: Labels must not obscure other labels or graphical features. A label must be easily identified with exactly one graphical feature. The position of each label must eliminate any ambiguity with respect to which graphical feature the label in question is associated with. A label must be placed in the best position, among all legible positions.

In the production of geographical maps, we rank label positions according to rules developed through years of experience with manual placement. These rules typically capture the aesthetic quality of label positions. For technical maps or drawings we must be able to customize the rules of label quality to meet a user's specific needs and/or expectations. Furthermore, when graphical features are associated with more than one label, then we must take into account how labels for the same graphical feature influence each other.

Next, we present some constraints that may be used to ensure that each label is unambiguous and easily read and recognized, when more than one label is associated with a graphical feature. These constraints can be divided into three general categories: (*i*) *proximity*, (*ii*) *partial order*, and (*iii*) *priority*. To illustrate the three sets of constraints we will use as an example the labeling of a single edge with two labels. Label $l_s$ is associated with the source node and label $l_t$ is associated with the target node.

**Proximity:** Label $l_s$ ($l_t$) must be in close proximity with the source (target) node to avoid ambiguity. Therefore, it is necessary to define a maximum distance from the source (target) node that label $l_s$ ($l_t$) must be positioned.

**Partial Order:** A label associated with the source (target) node must be closer to the source (target) node than the other label to avoid ambiguity. Thus, in many cases, it is appropriate to define a partial order among the labels according to some invariant (e.g., $x$ or $y$ axis, distance from a fixed point).

**Priority:** In many cases, it is impossible to assign all labels associated with a graphical feature due to the density of the drawing. Then, the user might prefer to have the important labels assigned first, and then assign the rest of the labels if there is available space.

## 3 Algorithms

**Algorithm 1**

First we present a simple iterative approach to solve the problem of assigning multiple labels to each graphical feature of a drawing. This method consists of a main loop, and we execute the loop as many times as the number of labels per graphical feature. For example, at the $i$-th execution of the loop, we assign the $i$-th label to each graphical feature. Labeling algorithms for single label per graphical feature may be used at each execution of the loop. This technique can take into account all three sets of constraints (proximity, partial order, priority). Thus, it leads to very practical solutions for the MLP problem. This iterative approach is especially suited for the labeling algorithms presented in [18, 19], because they first find a set of label positions, and then they produce a label assignment in a single step without any repositioning of labels.

**Algorithm 2**

Next we consider a technique for assigning label positions in a non-iterative fashion. The technique is an extension of the labeling algorithms presented in [18, 19]. It solves the MLP problem by reducing it to an assignment problem. First, a set of label positions for each graphical feature is constructed. Then, label positions that overlap each other are grouped into clusters. Next, each cluster of overlapping labels is matched to at most one graphical feature using flow techniques.

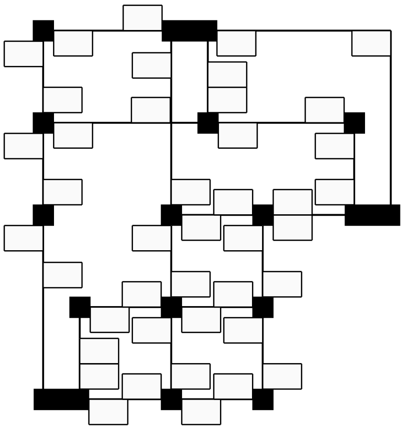

Figure 1: An orthogonal drawing with 2 labels per edge, positioned by Algorithm 1.

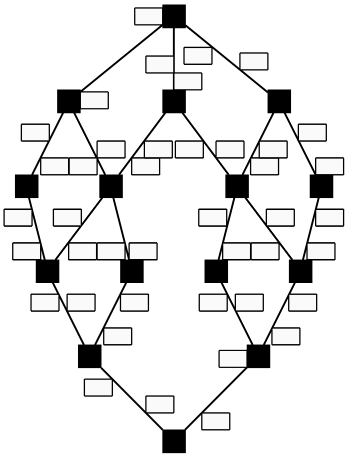

Figure 2: A hierarchical drawing with 2 labels per edge, positioned by Algorithm 1.

First we create a bipartite *matching* graph $G_m(V_f, V_c, E_m)$ in the following way:

- Each graphical feature to be labeled is represented by a node in $V_f$.
- Each group (cluster) of overlapping labels is represented by a node in $V_c$.
- Edges in $E_m$ connect each graphical feature to those clusters that contains label position(s) for that graphical feature.

Next we transform the matching graph $G_m$ to a flow graph $G_{flow}(s, t, V_f, V_c, E_f)$. $G_m$ is converted to an *st*-graph by introducing two nodes $s$ and $t$. Node $s$ is connected to each node in $V_f$, and node $t$ is connected to each node in $V_c$, as shown in Figure 3. Finally we assign capacities to each edge of the flow graph $G_{flow}$ in the following way:

- Each edge of the original matching graph has capacity one.
- Each edge $(c, t)$ of $G_{flow}$ incident to the target node has capacity one.
- Each edge $(s, v)$ incident to the source node has capacity equal to the number of labels associated with the graphical feature of the input graph that is represented by node $v$ in $G_m$.

Clearly a maximum flow of graph $G_{flow}$ will produce the maximum cardinality label assignment with respect to the set of labels encoded in the matching graph. Thus we have the following Theorem:

**Theorem 1** *A maximum flow of graph $G_{flow}$ produces a maximum cardinality label assignment with respect to the number of label positions encoded in the matching graph.* □

The key observation is that we require each label position of a cluster to overlap any other label position of the same cluster. We achieve that either by carefully constructing an initial set of label positions [18], or by reducing an initial set of label positions such that we construct a maximum number of minimum size clusters of overlapping labels [19].

We include the cost (with respect to quality) of label position $l$ of graphical feature $o$ as the weight of edge $(o, l)$ in the matching graph. We assign weight equal to zero to edges of $G_{flow}$ incident to the source and target nodes. Finally, we find a maximum cardinality minimum cost

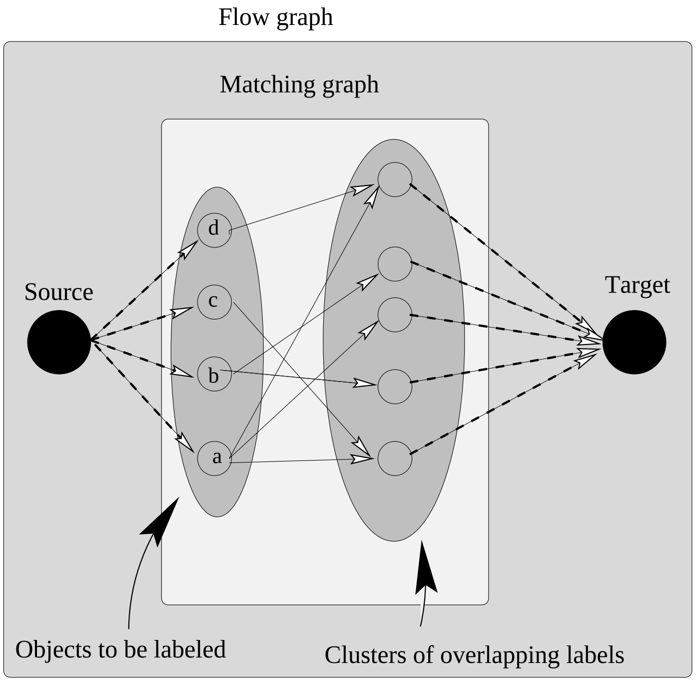


Figure 3: The flow graph of Algorithm 2.

label assignment for the MLP problem (with respect to the set of labels encoded in the matching graph) by solving the maximum flow minimum cost problem for the flow graph $G_{flow}$.

## 4 Experimental Results

We have implemented Algorithms 1 and 2 on top of the Graph Layout Toolkit[1] (GLT) in C++. We ran experiments on graphs taken from the set of graphs used in [5]. In our implementation the graphical features to be labeled are edges and nodes of graph drawings. We ran both algorithms on the same number of graphs with circular, hierarchical, orthogonal, and force-directed style drawings. All graphs have been drawn using the algorithms implemented in GLT [20]. We ran the experiments on a SPARC 5 unix based machine.

For the experiments, labels are rectangles and in successive runs of the algorithms we increase only the width of the labels. Furthermore, we ran experiments and collected data for one up to five labels per graphical feature. We were more interested on how the success rate (percentage of graphical features with assigned labels) was decreasing as we were increasing the size and the number of the labels. All drawings are grid drawings, that is nodes and bends have integer coordinates. The label size is with relation to the grid-size. For example, a label with width 20 is a label with width equal to 20 grid units.

We present some of the results in Figures 5 through 7. The plain lines represent the results after running Algorithm 1. The marked lines represent the results after running Algorithm 2. From these figures it is clear that our algorithms perform very well, since the success rate of assigning small labels is almost 100% and it drops only by a constant factor when we increase the label size and/or the number of labels per graphical feature. The algorithms run slower for smaller size labels because the initial set of label positions is much larger for smaller size labels. A

[1]GLT is a product of Tom Sawyer Software Inc. (www.tomsawyer.com).

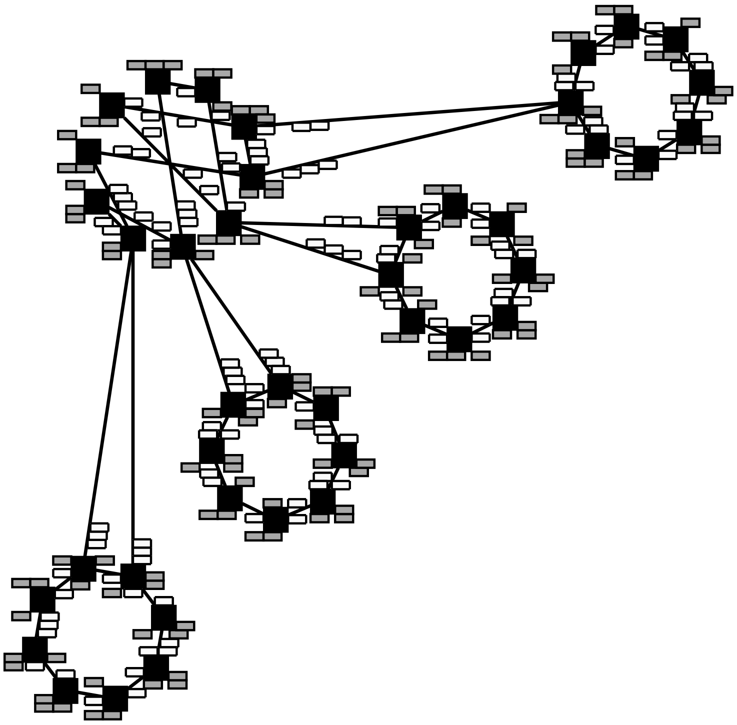

Figure 4: A circular drawing with three labels for each edge and node, positioned by Algorithm 2. The white boxes are edge labels and the dark boxes are node labels.

considerable amount of time is spent in detecting overlaps. We have implemented a simple plane sweep to detect overlaps. We could improve this part of our code by using the techniques in [8, 9] when label positions are parallel to the axis or the techniques of [15] that detect intersections between convex polygons. Both labeling algorithms perform essentially the same with respect to the success rate of assigning labels, even though the flow method produces a slightly better quality of label assignments. Finally, if labels for the same graphical feature have equal size, then Algorithm 2 is preferable since it is faster. If labels have variable size or strict partial order constraints, then Algorithm 1, which is more flexible, will be the appropriate choice.

# References


[1] J. Ahn and H. Freeman. A program for automatic name placement. *Cartographica*, 21(2 & 3):101–109, Summer & Autumn 1984.

[2] B. Chazelle et al. Application challenges to computational geometry: CG impact task force report. Technical Report TR-521-96, Princeton Univ., April 1996.

[3] 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.

[4] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Comput. Geom. Theory Appl.*, 4:235–282, 1994.

[5] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An Experimental Comparison of Four Graph Drawing Algorithms. *Comput. Geom. Theory Appl.*, 7(5-6):303–325, 1997.

[6] J. S. Doerschler and H. Freeman. A rule based system for dense map name placement. *Communications of ACM*, 35(1):68–79, 1992.

[7] L. R. Ebinger and A. M. Goulete. Noninteractive automated names placement for the 1990 decennial census. *Cartography and Geographic Informaton Systems*, 17(1):69–78, January 1990.

[8] H. Edelsbrunner. A new approach to rectangle intersections, Part I. *Internat. J. Comput. Math.*, 13:209–219, 1983.

[9] H. Edelsbrunner. A new approach to rectangle intersections, Part II. *Internat. J. Comput. Math.*, 13:221–229, 1983.

[10] S. Edmondson, J. Christensen, J. Marks, and S. Shieber. A General Cartographic Labeling Algorithm. *Cartographica*, 33(4):321–342, 1997.

[11] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 281–288, 1991.

[12] H. Freeman. An Expert System for the Automatic Placement of Names on a Geographic Map. *Information Sciences*, 45:367–378, 1988.

[13] H. Freeman. Computer name placement. In D. J. Maguire, M. F. Goodchild, and D. W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, pages 445–456. Longman, London, 1991.

[14] H. Freeman and J. Ahn. On the problem of placing names in a geographical map. *Int. J. of Pattern Rec. and Artificial Intelligence*, 1(1):121–140, 1987.

[15] P. Gupta, R. Janardan, and M. Smid. Efficient Algorithms for Counting and Reporting Pairwise Intersections between Convex Polygons. In *Proc. 8th Canadian Conference on Computational Geometry*, pages 8–13. Carleton University Press, 1996.

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

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

[18] K. G. Kakoulis and I. G. Tollis. An Algorithm for Labeling Edges of Hierarchical Drawings. In G. Di Battista, editor, *Graph Drawing (Proc. GD '97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 169–180. Springer-Verlag, 1998.

[19] K. G. Kakoulis and I. G. Tollis. A Unified Approach to Labeling Graphical Features. In *Proc. 14th Annual ACM Sympos. in Computational Geometry*, pages 347–356, June 1998.

[20] Tom Sawyer Software. Graph Layout Toolkit Reference Manual. Berkeley, CA, 1992-1998.

[21] F. Wagner and A. Wolff. Map labeling heuristics: Provably good and practically useful. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 109–118, 1995.

[22] P. Yoeli. The logic of automated map lettering. *The Cartographic Journal*, 9(2):99–108, 12 1972.

[23] S. Zoraster. The solution of large 0-1 integer programming problems encountered in automated cartography. *Operation Research*, 38(5):752–759, September-October 1990.

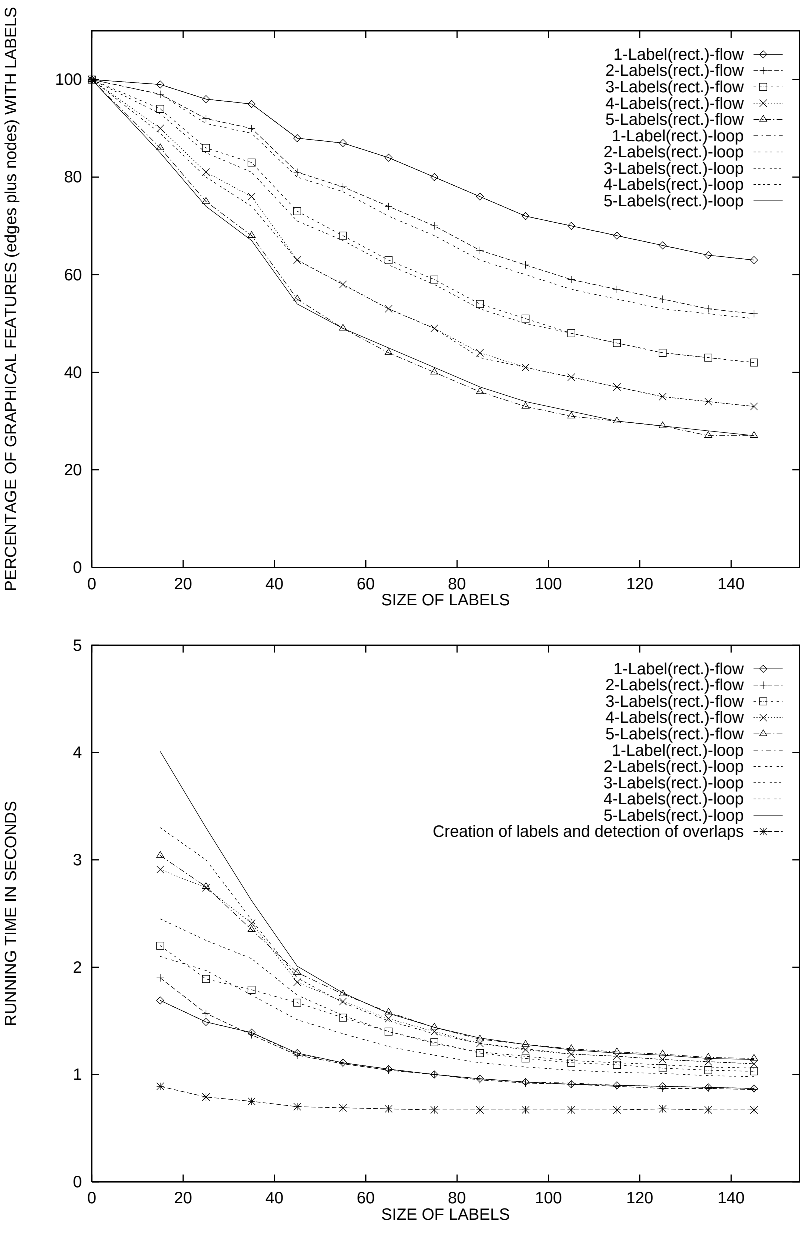


Figure 5: Experimental results for circular drawings of 30 nodes and on the average 36 edges with rectangular labels.

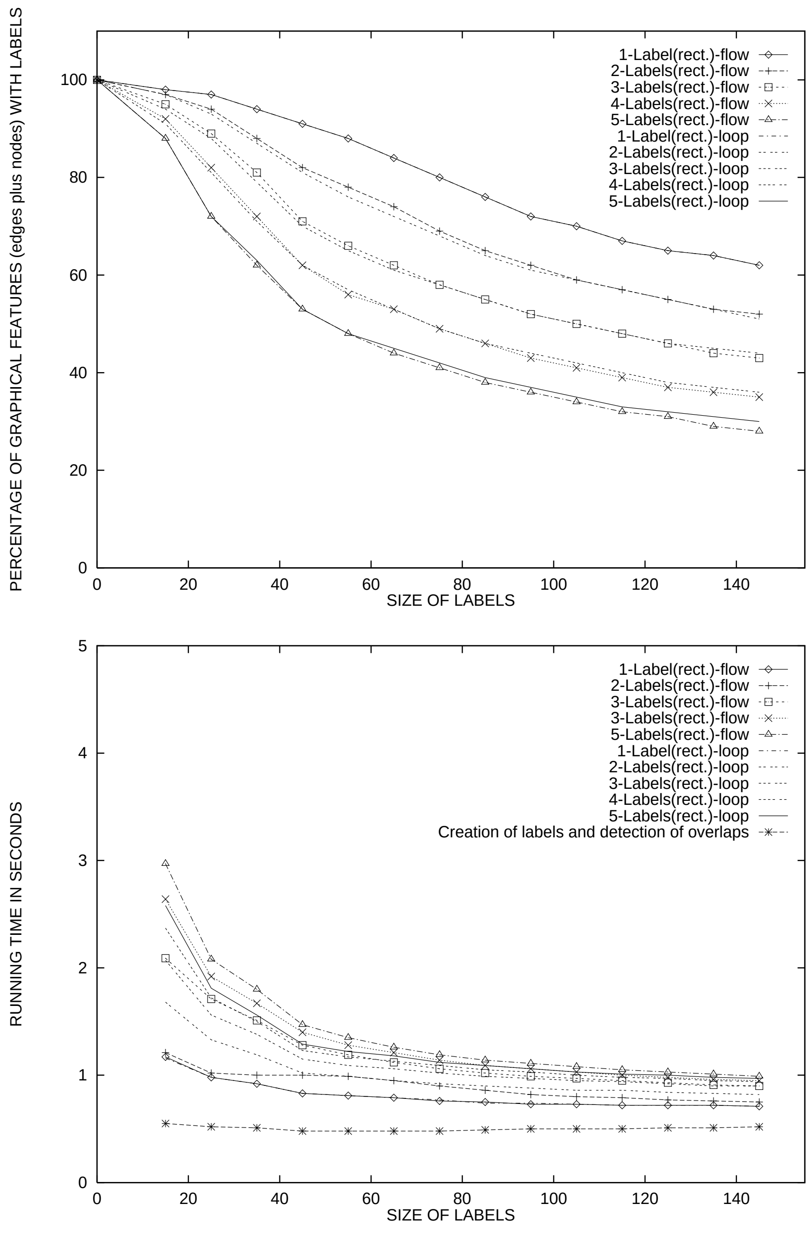


Figure 6: Experimental results for symmetric drawings of 30 nodes and on the average 36 edges with rectangular labels.

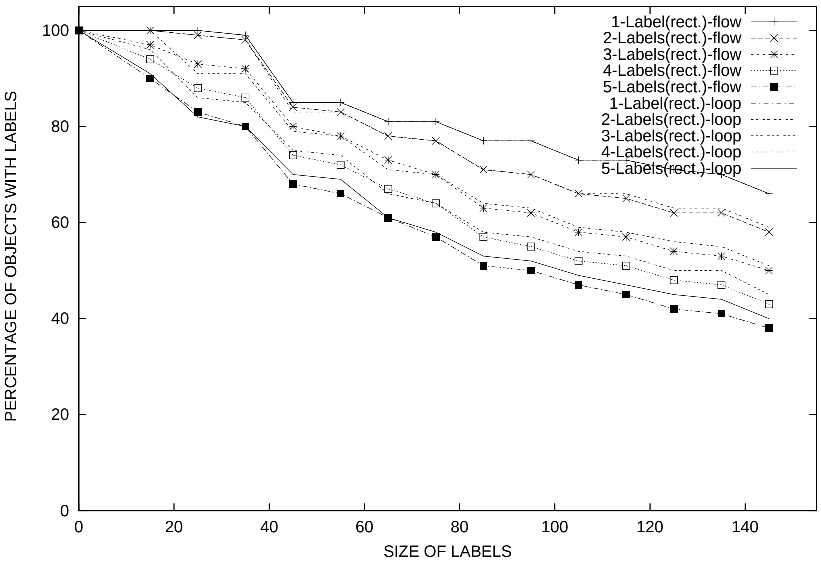


Figure 7: Experimental results for orthogonal drawings of 60 nodes and on the average 79 edges with rectangular labels.