Placing Arrows in Directed Graph Drawings

Carla Binucci[1(✉)], Markus Chimani[2], Walter Didimo[1], Giuseppe Liotta[1], and Fabrizio Montecchiani[1]

[1] Università degli Studi di Perugia, Perugia, Italy
{carla.binucci,walter.didimo,giuseppe.liotta, fabrizio.montecchiani}@unipg.it
[2] Osnabrück University, Osnabrück, Germany
markus.chimani@uni-osnabrueck.de

Abstract. We consider the problem of placing arrow heads in directed graph drawings without them overlapping other drawn objects. This gives drawings where edge directions can be deduced unambiguously. We show hardness of the problem, present exact and heuristic algorithms, and report on a practical study.

1 Introduction

The default way of drawing a directed edge is to draw it as a line with an arrow head at its target. While there also exist other models (placing arrows at the middle, drawing edges in a "tapered" fashion, etc.; cf. [7,8]) the former is prevailing in virtually all software systems. However, this simple model becomes problematic when several edges attach to a vertex on a similar trajectory: it may be hard to see whether a specific edge is in- or outgoing, cf. Fig. 1 and [1].

We try to solve this issue by looking for a placement of the arrow heads such that (a) they do not overlap other edges or arrow heads, and (b) still retain the property of being at—or at least close to—the target vertices of the edges. In the following, we show NP-hardness of the problem, propose exact and heuristic algorithms for its discretized variant, and evaluate their practical performance in a brief exploratory study. We remark that our problem is related to map labeling and in particular to edge labeling problems [4,5,9–11,13,15–18].

For space reasons, some proofs and technical details are omitted in this extended abstract, and can be found in the appendix of the ArXiv version [1].

2 The Arrow Placement Problem

We first formally define our arrow placement problem and establish its theoretical time complexity. Let $G = (V, E)$ be a digraph and let Γ be a straight-line drawing of G. We assume that in Γ each vertex $v \in V$ is drawn as a circle

Work is partially supported by the MIUR project AMANDA "Algorithmics for MAssive and Networked DAta", prot. 2012C4E3KT_001.

Y. Hu and M. Nöllenburg (Eds.): GD 2016, LNCS 9801, pp. 44–51, 2016.
DOI: 10.1007/978-3-319-50106-2_4

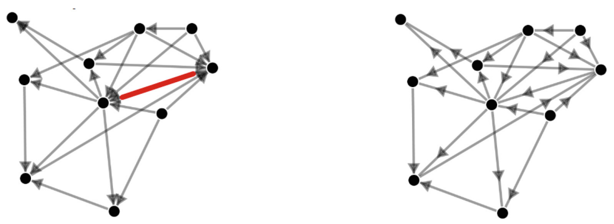


Fig. 1. Layouts of a digraph with 10 vertices and 21 edges. (left) The arrows are placed by a common editor; several arrows overlap and the direction of, e.g., the thick red edge is not clear. (right) The arrows are placed by our exact method. (Color figure online)

(possibly a point) C_v. We also assume that, for each edge $e \in E$, the arrow of e is modeled as a circle C_e of positive radius, centered in a point along the segment that represents e: when Γ is displayed, the arrow of e is drawn as a triangle inscribed in C_e, suitably rotated according to the direction of e. We assume that all circles representing a vertex (arrow) have a common radius r_V (r_E, respectively). We say that two arrows—or an arrow and a vertex—*overlap* if their corresponding circles intersect in two points. An arrow and an edge *overlap* if the segment representing the edge intersects the circle representing the arrow in two points. For the sake of simplicity, we reuse terms of theoretical concepts also for their visual representation: "arrow" and "vertex" also refer to their corresponding circle in Γ; "edge" also refers to its corresponding segment in Γ.

Definition 1. *Let a_e denote the arrow of an edge $e \in E$. A* valid position *for a_e in Γ is such that: (P1) for every vertex $v \in V$, a_e and v do not overlap; (P2) for every edge $g \in E$, $g \neq e$, a_e and g do not overlap. An assignment of a valid position to each arrow is called a* valid placement *of the arrows, denoted by P_Γ.*

Definition 2. *Given a valid placement P_Γ, the* overlap number *of P_Γ is the number of pairs of overlapping arrows, and is denoted as* $\text{ov}(P_\Gamma)$*.*

Given a straight-line drawing Γ of a digraph $G = (V, E)$, and constants r_V, r_E, we ask for a valid placement P_Γ of the arrows (if one exists) such that $\text{ov}(P_\Gamma)$ is minimum. This optimization problem is NP-hard; we prove this by showing the hardness of the following decision problem Arrow-Placement.

Problem: Arrow-Placement

Instance: $\langle G = (V, E), \Gamma, r_V, r_E \rangle$.

Question: Does there exist a valid placement P_Γ of the arrows with $\text{ov}(P_\Gamma) = 0$?

Theorem 1. *The* Arrow-Placement *problem is NP-hard.*

The proof of Theorem 1 uses a reduction from Planar 3-SAT [12], and is similar to those used in the context of edge labeling [9,13,16,18]. It yields an instance of Arrow-Placement where the search of a valid placement P_Γ with $\text{ov}(P_\Gamma) = 0$ can be restricted to a finite number of valid positions for each arrow. Hence, Arrow-Placement remains NP-hard even if we fix a finite set of positions for each arrow, and a valid placement with overlap number zero (if any) may only choose from these positions. As this variant of Arrow-Placement, which we call Discrete-Arrow-Placement, clearly belongs to NP, it is NP-complete.

3 Algorithms

We describe algorithms for the optimization version of Discrete-Arrow-Placement. We assume that a set of valid positions for each arrow is given, based on $\{\Gamma, r_V, r_E\}$, and look for a valid placement P_Γ that minimizes $\mathrm{ov}(P_\Gamma)$ over this set of positions. We give both an exact algorithm and two variants of a heuristic, which we experimentally compare in Sect. 4. Given an edge $e \in E$, let A_e denote the set of valid positions for the arrow of edge e, and let $A := \bigcup_{e \in E} A_e$ be the set of all valid positions. Our algorithms are based on an *arrow conflict graph* C_A, depending on A, Γ, and r_E. The positions A form the node set of C_A. Two positions are *conflicting*, and connected by an (undirected) edge in C_A, if they correspond to positions of different edges and the arrows would overlap when placed on these positions. Finding a valid placement P_Γ with $\mathrm{ov}(P_\Gamma) = 0$ means to select one element from each A_e such that they form an independent set in C_A. More general, finding a valid placement P_Γ with $\mathrm{ov}(P_\Gamma) = k$ $(k \geq 0)$ means to select one element from each A_e such that they induce a subgraph with k edges in C_A. Our exact algorithm minimizes k using an ILP formulation, while our heuristic adopts a greedy strategy. Both techniques try to minimize the distance of each arrow from its target vertex as a secondary objective. However, our algorithms can be easily adapted to privilege other positions (e.g., close to the source vertices, in the middle of the edges, etc.), or to consider bidirected edges.

ILP Formulation. For each position $p_e \in A_e$ of an edge $e = (v, u)$, we have a binary variable x_{p_e}. We define a distance $d(p_e) \in \{1, \ldots, |A_e|\}$, from p_e to u: $d(p_e) = 1$ $(d(p_e) = |A_e|)$ means that p_e is the position closest (farthest, respectively) to u. Let $E_A := E(C_A)$ be the pairs of conflicting positions. For every $(p_e, p_g) \in E_A$, we define a binary variable $y_{p_e p_g}$. The total number of variables is $O(|A|^2)$, and we write:

$$\min \sum_{(p_e,p_g) \in E_A} y_{p_e p_g} + \frac{1}{M} \cdot \sum_{e \in E} \sum_{p_e \in A_e} d(p_e) x_{p_e} \tag{1}$$

$$\sum_{p_e \in A_e} x_{p_e} = 1 \qquad \forall e \in E \tag{2}$$

$$x_{p_e} + x_{p_g} \leq y_{p_e p_g} + 1 \qquad \forall (p_e, p_g) \in E_A \tag{3}$$

The objective function minimizes the overlap number and, secondly, the sum of the distances of the arrows from their target vertices. To do this, the second term is divided for a sufficiently large constant M. For example, one can set $M = |E| \max_{e \in E}\{|A_e|\}$. Equation 2 guarantee that exactly one valid position per edge is selected. Constraint 3 enforces $y_{p_e p_g} = 1$ if both conflicting positions x_{p_e} and x_{p_g} are chosen. In the following, the exact technique will be referred to as Opt. We remark that optimization problems and ILP formulations similar to above have been given in the context of edge and map labeling [4,5,9,11,15,16].

Heuristics. Our heuristics follow a greedy strategy, again based on C_A. Let $p_e \in A_e \subset V(C_A)$ as above. We initially assigns cost $c(p_e)$ to each position p_e, and then execute $|E|$ iterations. In each iteration, we select a position p_e of minimum cost (over all $e \in E$) and place the arrow of the corresponding edge there; then, we remove all positions A_e from C_A (including p_e), and update the costs of the remaining positions. We define $c(p_e) := \delta(p_e) + \frac{1}{M} d(p_e) + T\sigma_{p_e}$, where: $\delta(p_e)$ is the degree of p_e in C_A (i.e., the number of positions conflicting with p_e); constant M and "distance" $d(p_e)$ are defined as in the ILP; σ_{p_e} is the number of already chosen positions conflicting with p_e (0 in the first iteration); T is equal to the maximum initial cost of a valid position. This cost function guarantees that: (i) positions conflicting with already selected positions are chosen only if necessary; (ii) the algorithm prefers positions with the minimum number of conflicts with the remaining positions and, among them, those closer to the target vertex. Since constructing C_A may be time-consuming in practice (we compare all pairs of valid positions), we also consider using only a subset of the edges of C_A; we may consider only those conflicts arising from positions of adjacent edges in the input graph. In the following, HeurGlobal is the heuristic that considers full C_A, while HeurLocal is the variant based on this simplified version of C_A.

4 Experimental Analysis

Experimental Setting. We use three different sets of graph: Planar are biconnected planar digraphs with edge density 1.5–2.5, randomly generated with the OGDF [3]. Random are digraphs generated with uniform probability distribution with edge density 1.4–1.6. Both sets contain 30 instances each; 6 graphs for each number of vertices $n \in \{100, 200, \ldots, 500\}$. We did not generate denser graphs, as they give rise to cluttered drawings with few valid positions for the arrows—there, the arrow placement problem seems less relevant. Finally, North is a popular set of $1,275$ real-world digraphs with 10–100 vertices and average density 1.4 [14]. We draw each instance of the three sets with straight-line edges using OGDF's FM3 algorithm [6]. The layouts of the Planar may contain edge crossings, as they are generated by a force-directed approach.

Value r_E is chosen as the minimum of (a) 40% of the shortest edge length, (b) 25% of the average edge length, and (c) 10 pixels, but enforced to be at least 3 pixels. We set $r_V := r_E$. For each edge $e = (w, u)$ we compute positions A_e as follows. The i-th position, $i \geq 1$, has its center at distance $r_V + i \cdot r_E$ from target u. We generate positions as long as they have distance at least $r_V + r_E$ from source vertex w. We then remove positions that overlap with edges or vertices in Γ. If no valid positions remain, we choose the one closest to u as e's unique arrow position. Thus, in the final placements there might be some conflicts between an arrow and a vertex or edge of the drawing. We call such conflicts *crossings* and observe that a single invalid position may result in several crossings.

We apply Opt, HeurGlobal, and HeurLocal to each of the drawings. The algorithms are implemented in C# and run on an Intel Core i7-3630QM notebook with 8 GB RAM under Windows 10. For the ILP we use CPLEX 12.6.1 with default settings. For each computation, we measure total running time, overlap number, and number of crossings (due to invalid positions, see above). From the qualitative point of view, we also compare the algorithmic results with a trivial placement, called Editor, which simply places each arrow close to its target vertex, similarly as most graph editors do. We also measure *placement time*, i.e., the time spent by an algorithm to find a placement *after* C_A has been computed.

Results. For PLANAR, the average numbers of positions in C_A range from 640 to $7,150$. Figures 2(a) and (b) show that for PLANAR all the algorithms are very applicable, although Opt is of course significantly slower. While the pure placement time for HeurGlobal is not much longer than that of HeurLocal, it suffers from the fact that generating the full C_A constitutes roughly 1/3 of its overall runtime, whereas the generation time of the reduced conflict graph is rather neglectable. On the other hand, Fig. 2(c) shows that HeurGlobal practically coincides with the optimum w.r.t. the number of overlaps (its average gap is below 3%; the worst gap is 6.76%). HeurLocal still gives very good solutions, with gaps about half that of Editor. Figure 2(d) shows that our algorithms reduce the number of invalid positions by $33-77\%$ compared to Editor. The number of

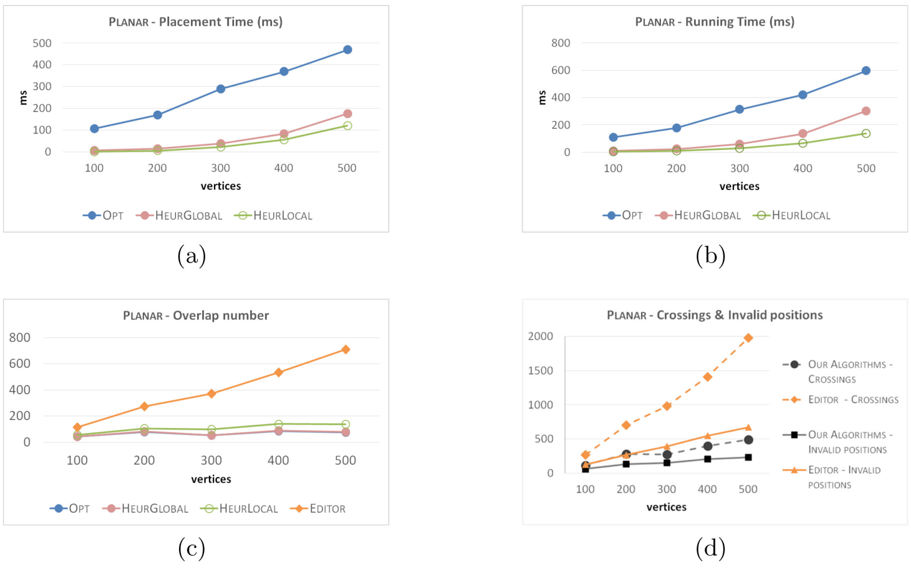
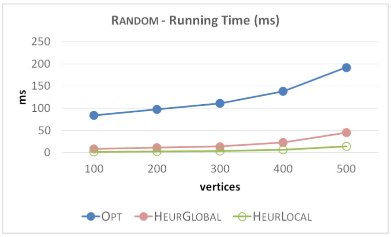
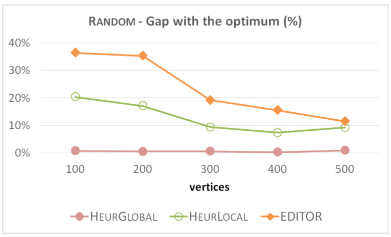


Fig. 2. PLANAR: (a) Placement time; (b) Total running time; (c) Number of overlaps; (d) Number of crossings (edge/vertex with arrow) and of invalid positions.

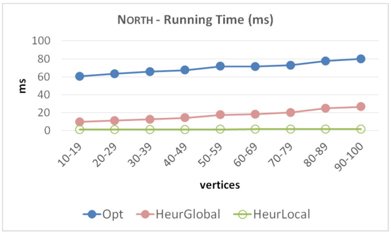


(a)

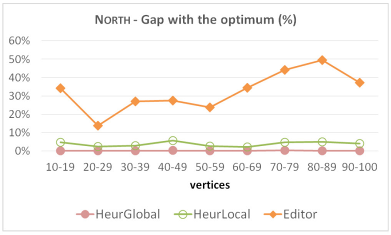


(b)

Fig. 3. Random: (a) Total running time; (b) Number of overlaps, relative to Opt.



(a)



(b)

Fig. 4. North: (a) Total running time; (b) Number of overlaps, relative to to Opt.

crossings is the same for all our algorithms, as they occur when we cannot find any valid position for arrows during the generation procedure. Figure 2(d) shows that our algorithms cause significantly less crossings than Editor.

For Random, average numbers of positions in C_A range from 640 to 4,377. The general behavior for Random is similar to that of Planar but the difference between the running time of Opt and the heuristics is slightly more pronounced (Fig. 3(a)). Again, constructing C_A constitutes roughly 1/3 of HeurGlobal's running time. Still, the quality of HeurGlobal's solutions again essentially coincide with Opt; the other heuristics are now closer than before, see Fig. 3(b).

For North, the average $|V(C_A)|$ range from 62 to 311. We observe the same patterns, see Figs. 4: HeurLocal requires nearly no time, while HeurGlobal is very competitive at just above 20ms for the large graphs (a third of which is the construction of full C_A). Again, Opt always finds a solution very quickly, in fact within roughly 80ms. HeurGlobal again gives essentially optimal solutions, while HeurLocal exhibits 5–10% gaps. Editor requires 30–50% more overlaps than Opt.

5 Conclusions and Future Work

We discussed optimizing arrow head placement in directed graph drawings, to improve readability. As mentioned, this is very related to studies in map and graph labeling, but its specifics seem to make a more focused study worthwhile.

Our techniques are of practical use, and could be sped-up by constructing C_A using a sweepline or the labeling techniques in [17]. It would be interesting to validate the effectiveness of our approach through a user study (e.g. for tasks that involve path recognition). Moreover, one may consider both placing labels and arrow heads. Finally, the non-discretized problem variant, as well as the variants' respective (practical) benefits, should be investigated in more depth.

Acknowledgments. Research on this problem started at the Dagstuhl seminar 15052 [2]. We thank Michael Kaufmann and Dorothea Wagner for valuable discussions, and the anonymous referees for their comments and suggestions.

References

1. Binucci, C., Chimani, M., Didimo, W., Liotta, G., Montecchiani, F.: Placing arrows in directed graph drawings. ArXiv e-prints abs/1608.08505 (2016). http://arxiv.org/abs/1608.08505v1
2. Brandes, U., Finocchi, I., Nöllenburg, M., Quigley, A.: Empirical evaluation for graph drawing (Dagstuhl seminar 15052). Dagstuhl Rep. **5**(1), 243–258 (2015)
3. Chimani, M., Gutwenger, C., Jünger, M., Klau, G.W., Klein, K., Mutzel, P.: The open graph drawing framework (OGDF). In: Tamassia, R. (ed.) Handbook of Graph Drawing and Visualization, chap. 17. CRC Press, Boca Raton (2014). www.ogdf.net
4. Gemsa, A., Niedermann, B., Nöllenburg, M.: Trajectory-based dynamic map labeling. In: Cai, L., Cheng, S.-W., Lam, T.-W. (eds.) ISAAC 2013. LNCS, vol. 8283, pp. 413–423. Springer, Heidelberg (2013). doi:10.1007/978-3-642-45030-3_39
5. Gemsa, A., Nöllenburg, M., Rutter, I.: Evaluation of labeling strategies for rotating maps. In: Gudmundsson, J., Katajainen, J. (eds.) SEA 2014. LNCS, vol. 8504, pp. 235–246. Springer, Heidelberg (2014). doi:10.1007/978-3-319-07959-2_20
6. Hachul, S., Jünger, M.: Drawing large graphs with a potential-field-based multilevel algorithm. In: Pach, J. (ed.) GD 2004. LNCS, vol. 3383, pp. 285–295. Springer, Heidelberg (2005). doi:10.1007/978-3-540-31843-9_29
7. Holten, D., Isenberg, P., van Wijk, J.J., Fekete, J.: An extended evaluation of the readability of tapered, animated, and textured directed-edge representations in node-link graphs. In: IEEE PacificVis 2011, pp. 195–202. IEEE (2011)
8. Holten, D., van Wijk, J.J.: A user study on visualizing directed edges in graphs. In: CHI 2009, pp. 2299–2308. ACM (2009)
9. Kakoulis, K.G., Tollis, I.G.: On the complexity of the edge label placement problem. Comput. Geom. **18**(1), 1–17 (2001)
10. Kakoulis, K.G., Tollis, I.G.: Labeling algorithms. In: Tamassia, R. (ed.) Handbook on Graph Drawing and Visualization, pp. 489–515. Chapman and Hall/CRC, New York (2013)
11. van Kreveld, M.J., Strijk, T., Wolff, A.: Point labeling with sliding labels. Comput. Geom. **13**(1), 21–47 (1999)
12. Lichtenstein, D.: Planar formulae and their uses. SIAM J. Comput. **11**(2), 329–343 (1982)
13. Marks, J., Shieber, S.: The computational complexity of cartographic label placement. Technical Report 05-91, Harvard University (1991)
14. North graphs. http://www.graphdrawing.org/data.html

15. Strijk, T., van Kreveld, M.J.: Practical extensions of point labeling in the slider model. GeoInformatica **6**(2), 181–197 (2002)
16. Strijk, T., Wolff, A.: Labeling points with circles. Int. J. Comput. Geom. Appl. **11**(2), 181–195 (2001)
17. Wagner, F., Wolff, A., Kapoor, V., Strijk, T.: Three rules suffice for good label placement. Algorithmica **30**(2), 334–349 (2001)
18. Wolff, A.: A simple proof for the NP-hardness of edge labeling. Technical Report 11/2000, Institute of Mathematics and Computer Science, Ernst Moritz Arndt University Greifswald (2000)