Optimizing a Radial Layout of Bipartite Graphs for a Tool Visualizing Security Alerts

Maxime Dumas[1], Michael J. McGuffin[1], Jean-Marc Robert[1], and Marie-Claire Willig[2]

[1] ETS
Montréal, Canada
{michael.mcguffin,jean-marc.robert}@etsmtl.ca
[2] EESTIN, UHP Nancy 1
Vandoeuvre Lès Nancy, France

Abstract. Effective tools are crucial for visualizing large quantities of information. While developing these tools, numerous graph drawing problems emerge. We present solutions for reducing clutter in a radial visualization of a bipartite graph representing the alerts generated by an IDS protecting a computer network. Our solutions rely essentially on (i) unambiguous edge bundling to reduce the number of edges to display and (ii) the minimization of the total sum of the edge lengths.

Keywords: IDS alerts, bipartite graph layout, edge bundling.

1 Introduction

Intrusion Detection Systems (IDSs) are important tools for protecting enterprise networks. Unfortunately, they generate large quantities of information that are challenging to analyze. Few visualization tools have been proposed to ease the effort of network defence analysts [1,5]. These tools either do not use any graph layouts or do not focus on optimizing graph layouts.

Recently, the authors developed a new tool for IDS visualization named AlertWheel [4] that employs a radial overview visualization with a novel form of edge bundling, and incorporates features for filtering and drilling down on IDS alerts. IDSs such as SNORT [7] generate alerts when abnormal traffic flows are detected. The information in each alert identifies the category of the malicious behaviour (e.g., *network-scan*, *web-application-attack*, etc.) and the origin of the flow (the source IP address of the packets, from which an *Autonomous System* (AS) *Number* can be computed). Hence, these alerts can be visualized as the edges of a bipartite graph, where each node is either the AS node of the source, or the alert category. (The use of AS rather than IP addresses greatly reduces the number of source nodes in the bipartite graph.)

AlertWheel relies on a new way of drawing bipartite graphs that is visually clearer than the status quo (see Fig. 1). The inner circle corresponds to a limited number (up to 32) of alert categories, and the outer circle corresponds to AS nodes (see Fig. 5 and 7). In the development of this tool, multiple graph drawing

M. van Kreveld and B. Speckmann (Eds.): GD 2011, LNCS 7034, pp. 203–214, 2012.

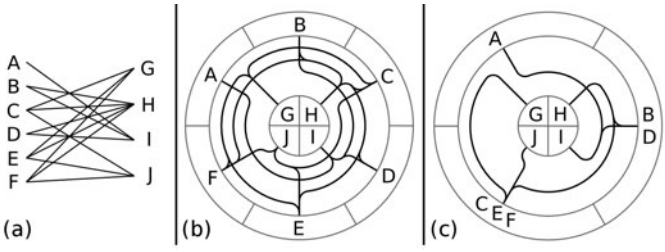

Fig. 1. The same bipartite graph shown in different ways. (a) Status-quo approach. (b) An improved way of drawing edges: each edge starting on an outer node leads to a circular bundle, and each circular bundle leads to a single inner node. (c) A further improvement that groups together nodes having the same neighbors.

problems were encountered. The aim of this paper is to present these abstract problems and the heuristics used to solve them. The companion paper [4] focuses on the overall tool and its interactive features.

2 Problem Statement

AlertWheel displays a radial visualization of a bipartite graph composed of an inner circle on which there are n interior points $i_1, \cdots, i_n$ (representing categories) and an outer circle on which there are m exterior points $o_1, \cdots, o_m$ (representing AS source nodes). Each edge between an interior and exterior point represents an observed alert. The positions of the interior points are determined by a central pie chart (Fig. 5 and 7). The interior points are sorted such that i_1 is connected to the minimum number of exterior points and i_n is connected to the maximum.

Our objective is to layout the numerous edges connecting the points as efficiently as possible, to ease reading and interpretation. This is done by

- Grouping edges into bundles [6,8];
- Reducing the sum of the edge lengths.

The edge bundling is illustrated in Fig. 1 and 2. Edges are layered on concentric circles where each concentric circle corresponds to one interior point. Edges can share either radial or circular segments with other edges, without introducing any ambiguity. Grouping together nodes with the same neighbors (Fig. 1c) further reduces clutter, and is very useful for visualizing security alerts. During outbreaks of very virulent malware, many infected computers may knock at the door of a given network and generate alerts of the same categories.

In the development of AlertWheel, the following graph drawing problems had to be addressed:

Problem I. Choose an assignment of concentric circles to interior points.

Problem II. Choose the position of a single exterior point minimizing the sum of lengths of edges to its neighbors.

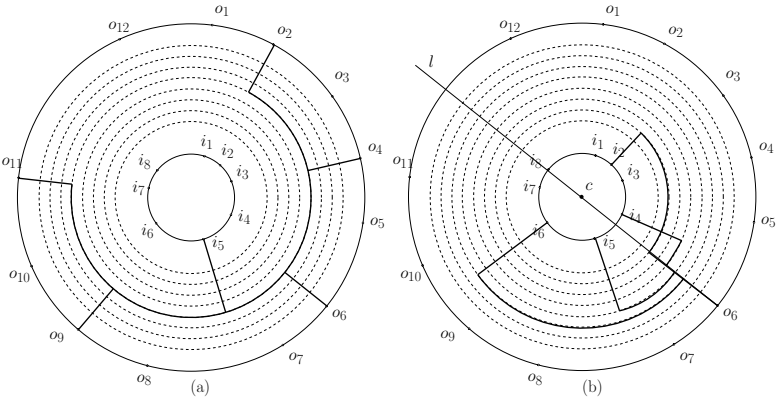

Fig. 2. Edge bundling. (a) Connecting multiple exterior points to one interior point – e.g., computers from five AS nodes generating alerts of the same category. (b) Connecting multiple interior points to one exterior point – e.g., a single AS node generating alerts of four categories.

Problem III. Choose the positions of all exterior points minimizing the total sum of the edge lengths.

In the next sections, solutions to these problems are presented. Exact solutions are only given for the first two problems, and heuristics are given for the last one. These heuristics are compared with each other and with the non-achievable lower bound derived from the second problem. Problem III is the core layout problem in AlertWheel. Efficient solutions to this problem are thus crucial.

3 Related Work

Network defence analysts continuously monitor their computer networks to detect any malicious activities. They have to analyze the data buried in numerous log files. Without appropriate tools, they cannot manage all this information.

The ultimate goal for security visualization tools is to present the data as simply as possible. To achieve this ambitious objective, numerous graph drawing problems have to be addressed [10]. These generally involve optimizing some aesthetic properties of the graph layout such as the number of edge crossings, the number of edge bends, etc.

Purchase [9] asserts that minimizing the number of edge crossings in a graph layout is the most important issue to deal with. Unfortunately, numerous variants of this problem have been shown to be NP-complete [11,2]. Edge bundling can be used to mitigate the effects of edge crossings. By merging edges into bundles [6,8], the number of edge crossings is significantly reduced. In such a case, reducing the length of the bundled edges remains an important goal. The rest of this paper addresses this goal.

4 Assignment of Concentric Circles to Interior Points

Concentric circles are used to layout the edges of the bipartite graph. These are particularly useful to bundle the edges connected to a given interior point (as in Fig. 2a). Assume that some set of radii r_k, $k = 1, \ldots, n$ of concentric circles has been chosen, for example with equal spacing. The first problem is to find an optimal assignment of each radius to a unique interior point, i.e., an assignment minimizing the sum of the edge lengths. Unfortunately, the total length of edges depends on the assignment of points on the outer circle. Thus, in this section, we simplify the problem and seek an assignment of radii to interior points that minimizes the total length of the *radial* components of the edges.

Let E_k be the subset of exterior points connected to an interior point i_k using the concentric circle with radius r_k (as in Fig. 2a) and let e_k be its cardinality. The sum of the radial edge lengths l_k^R is given by $e_k(r_{n+1} - r_k) + (r_k - r_0)$ where r_0 and r_{n+1} are the radii of the inner and outer circles, respectively.

The following lemma shows that the optimal assignment depends only on the number of edges using each circle. It can be proved easily by contradiction.

Lemma 1. *Suppose that the numbers of edges e_k are such that $e_1 \leq \cdots \leq e_n$. If the radii are such that $r_1 \leq \cdots \leq r_n$, then $S^R = \sum_{k=1}^{n} l_k^R$ is minimum.*

5 Optimally Connecting an Exterior Point to Interior Ones

The next problem is to find the optimal position of an exterior point to minimize the sum of its edge lengths. We assume that the assignment of radii is fixed and given by Lemma 1. This leaves only the *circular* components to be minimized. (Note that this heuristic does not guarantee that the *total* length of edges is minimized.) Then, the objective can be restated as finding the optimal position of an exterior point minimizing the total length of the circular components of its edges, given the assumed assignment of radii to interior points.

Consider an exterior point o connected to k interior points $i_1, i_2, \cdots, i_k$ (as in Fig. 2b). Let l be the line passing through the center c of the circles and the exterior point o. This line partitions the interior points into three sets: the points lying above l ($\mathcal{A}$), the points lying below l ($\mathcal{B}$) and the points lying on l ($\mathcal{O}$).

Let Θ_j be the angle in radians defined by the points o, c and i_j, defined s.t. $0 \leq \Theta_j \leq \pi$. The sum of the circular edge components is

$$S^C = \sum_{i_j \in \mathcal{A}} \Theta_j \cdot r_j + \sum_{i_j \in \mathcal{B}} \Theta_j \cdot r_j + \sum_{i_j \in \mathcal{O}} \Theta_j \cdot r_j \tag{1}$$

The following lemma characterizes the optimal solutions minimizing the sum of the circular edge components for a given exterior point o.

Lemma 2. *There is an optimal position for the exterior point o minimizing the sum of the circular edge components s.t. the line l defined by c and o passes through an interior point i_j between c and o.*

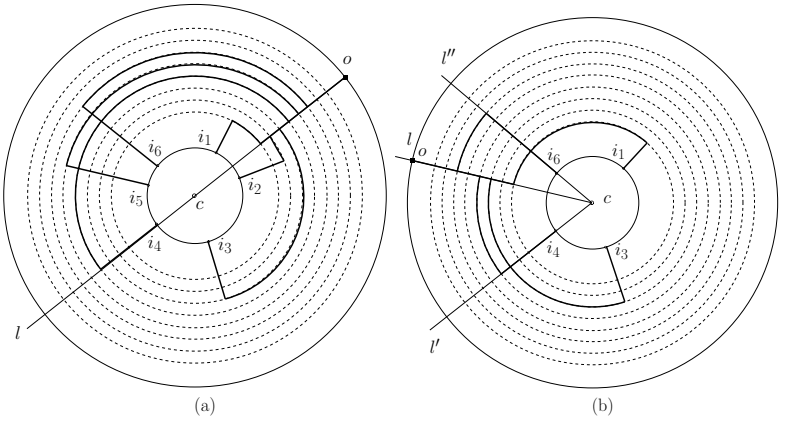

Fig. 3. Optimally positioning an exterior point connected to interior points

Proof. Suppose there is an optimal position for o which does not satisfy the criteria as in Fig. 3a. (Note that, in this case, there could still be a point in $\mathcal{O}$, located on the "other side" of o (Fig. 3a).) Let S^C be the corresponding sum of the lengths as given by Eq. 1. W.l.o.g. suppose that $\sum_{i_j \in \mathcal{A}} r_j > \sum_{i_j \in \mathcal{B}} r_j$ as in Fig. 3a. By rotating l counter-clockwise by a small angle $\epsilon > 0$, the point in $\mathcal{O}$ (if it exists) would be located above l. The sum of lengths would then be

$$S' = \sum_{i_j \in \mathcal{A}} (\Theta_j - \epsilon) \cdot r_j + \sum_{i_j \in \mathcal{B}} (\Theta_j + \epsilon) \cdot r_j + \sum_{i_j \in \mathcal{O}} (\Theta_j - \epsilon) \cdot r_j$$

$$= S^C - \epsilon \left[\sum_{i_j \in \mathcal{A} \cup \mathcal{O}} r_j - \sum_{i_j \in \mathcal{B}} r_j \right] < S^C.$$

This contradicts the optimality hypothesis of S^C.

In the special case $\sum_{i_j \in \mathcal{A}} r_j = \sum_{i_j \in \mathcal{B}} r_j$ and $\mathcal{O} = \emptyset$, the line l can still be rotated onto either of two interior points (yielding l' and l'') without worsening the sum of lengths (Fig. 3b). □

This lemma gives a straightforward linear time algorithm to find an optimal solution once the interior points have been fixed and assigned radii. The algorithm simply has to sweep through the finite number of candidate solutions.

As the point o moves around the circle, the sum of the circular edge lengths can reach numerous local minima (Fig. 4). Hence, binary search algorithms based solely on local decisions could lead to non-optimal solutions.

6 Connecting Multiple Exterior Points to Multiple Interior Points

The last problem to consider is choosing the positions of exterior points minimizing the total sum of the circular edge lengths. This is the core layout problem

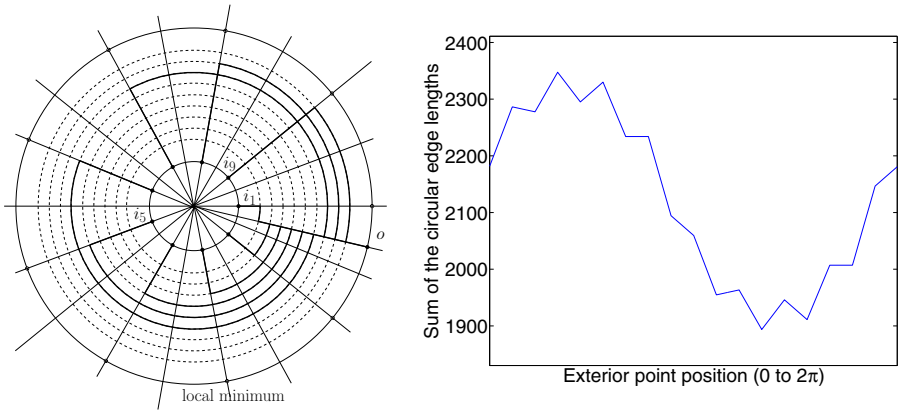

Fig. 4. Multi-modality of the function giving the sum of lengths. (Left) The construction of an example with 9 points – the radius r_k is given by $r_0+\delta+(k-1)\times\frac{r_{n+1}-r_0-2\delta}{n-1}$. (Right) The corresponding sum of the circular edge lengths.

of the AlertWheel visualization tool. Due to the nature of the problem, the positions of the interior points are fixed. Hence, changing the positions of exterior points is the only way to optimize the layout.

Two heuristics are presented to solve this optimization problem. These heuristics are compared with the naive solution of ordering the points on a first-come, first-served basis. A lower bound can be derived from the algorithm presented in the previous section. However, this solution may not be achievable since it allows many exterior points to coincide.

Let us first introduce some notation. Let $I_k \subseteq \{i_1, \cdots, i_n\}$ be the subset of the interior points connected to the exterior point o_k. These points represent the *hyperedge* associated with o_k. Also, let $\mathcal{I} = \{I_k | 1 \leq k \leq m\}$ be the set of the hyperedges to be laid out.

6.1 Heuristic I: The Minimum Perfect Matching

The first heuristic is an algorithm distributing the exterior points evenly on the outer circle. The eases point labeling in the visualization tool but does not guarantee that the obtained optimal solution is globally optimal.

The algorithm is based on the minimum perfect matching problem [3]. It constructs a complete bipartite graph $K_{m,m}$ representing the cost of associating each hyperedge I_k to each potential layout position. A minimum-weight perfect matching would give a one-to-one correspondence between the set of hyperedges and the set of layout positions which minimizes the total sum of the hyperedge lengths.

A more formal description of this heuristic is presented in Algorithm 1. The running time of this algorithm is $O(m^2n + m^3) = O(m^3)$, assuming $n < m$.

Let $\mathcal{I} = \{I_k | 1 \leq k \leq m\}$ be the hyperedges associated to the m exterior points.
Let $\mathcal{P} = \{p_k | 1 \leq k \leq m\}$ be a set of evenly distributed positions on the outer circle.
for $i \leftarrow 1$ **to** m **do**
    for $j \leftarrow 1$ **to** m **do**
        Compute the length $w_{i,j}$ of the hyperedge I_i at the position p_j.
Compute the minimum perfect matching of the complete bipartite graph defined by the sets $\mathcal{I}$, $\mathcal{P}$ and $\mathcal{W} = \{w_{i,j} | 1 \leq i, j \leq m\}$

Algorithm 1. Minimum perfect matching heuristic

In Fig. 5, the performance of this algorithm is compared with the naive algorithm ordering the points on a first-come, first-served basis. As expected, the heuristic yields a better result on this example than the naive algorithm. A more thorough comparison is presented in Section 6.3.

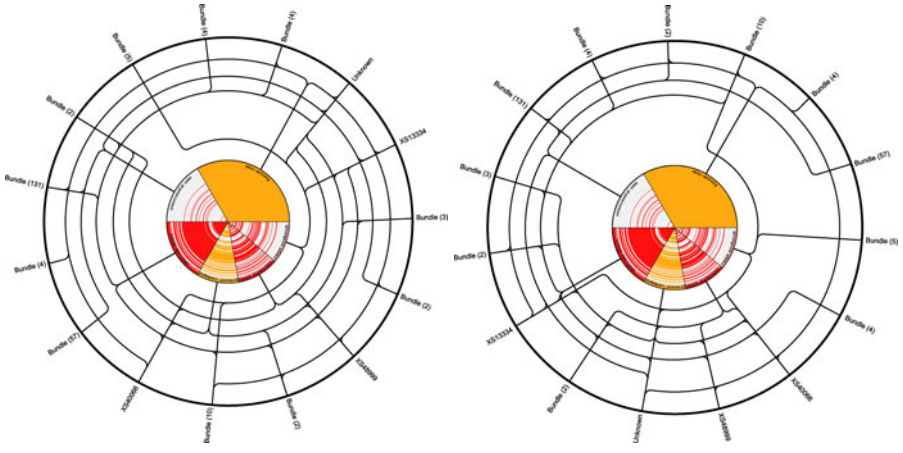

Fig. 5. AlertWheel tool: Comparing the naive algorithm (left) and the perfect matching heuristic (right). In the former case, the sum of the circular edge lengths is 22036 units. In the latter case, the sum is 17158, or 22% less.

6.2 Heuristic II: The Anchor Algorithm

The second heuristic is based on the idea of finding the optimal position of each exterior point and placing it as close as possible to this position, which we call an anchor. Lemma 2 determines the anchors for the exterior points. Based on the computed anchors, the exterior points are partitioned into sets containing points competing for the same optimal anchors. Then, the optimal local positioning of the exterior points around each anchor is determined.

Unfortunately, this algorithm does not guarantee an optimal solution. If an exterior point is placed too far from its optimal anchor, it could be better off at another anchor. Nevertheless, this heuristic yields good results, as we will see.

Let us introduce some notation. Let $C_p \subseteq \{o_1, \cdots, o_m\}$ be the points competing for the anchor p. To simplify the notation, assume that $C_p = \{o_1, \cdots, o_t\}$. The anchor p and center c determine a line l that divides the interior points into the points lying above l ($\mathcal{A}$), the points lying below l ($\mathcal{B}$) and the points lying on l. To simply the argument, we assume that only the anchor point lies on l. For any exterior point $o_i \in C_p$ connected to the hyperedge I_i, we define

$$a_i = \sum_{\substack{i_j \in I_i \\ i_j \in \mathcal{A}}} r_j \quad \text{and} \quad b_i = \sum_{\substack{i_j \in I_i \\ i_j \in \mathcal{B}}} r_j.$$

Finally, let r^* be the radius associated with the interior point defining the anchor p. By Lem. 2, this interior point must be in I_i.

Let $A = \{o_i \in C_p | a_i \geq b_i\}$ and let $B = \{o_i \in C_p | a_i < b_i\}$. Now, suppose that $o_i \in A$. As the exterior point o_i moves away from the anchor p by an angle $\Theta > 0$, the total sum of the circular edge lengths increased by

$$\begin{aligned} &(a_i + r^* - b_i)\Theta > 0 \qquad \text{if } o_i \text{ is moving away clockwise} \\ &(b_i + r^* - a_i)\Theta > 0 \qquad \text{if } o_i \text{ is moving away counter} - \text{clockwise.} \end{aligned}$$

Both expressions must be positive. Otherwise, the point p would not represent an optimal anchor for o_i. This follows from Lem. 2.

The following lemma characterizes the optimal layouts of the exterior points in A. Intuitively, these points should be moved away from the anchor counter-clockwise to reduce the impact on the sum of the circular edge lengths.

Lemma 3. *Let $0 \leq i \leq |A|$. There are only two optimal layouts of the exterior points in A around the anchor p s.t. i points of A move away clockwise.*

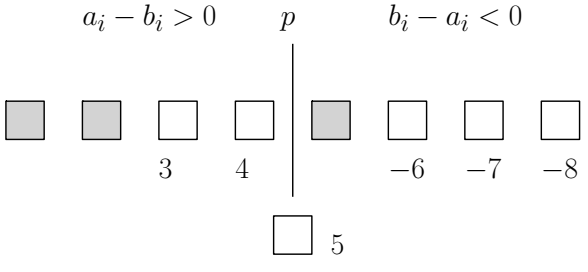

Fig. 6. Ordered layout of the exterior points in A for $i = 2$. The values represent the values of $a_i - b_i > 0$ (on the left of p) and the values of $b_i - a_i < 0$ (on the right of p). The shaded boxes represent the positions of the points in B.

Proof. W.l.o.g., suppose that $A = \{o_1, \cdots, o_r\}$ is such that the values of $a_i - b_i$ are sorted in increasing order. In order to minimize the total sum of the circular edge lengths, the points must be laid out as follows:

- $o_1, \cdots, o_i$ clockwise w.r.t. the anchor and $o_{i+1}, \cdots, o_r$ counter-clockwise w.r.t. the anchor – and according to the sorted order;
- $o_1, \cdots, o_i$ clockwise w.r.t. the anchor, o_{i+1} on the anchor and $o_{i+2}, \cdots, o_r$ counter-clockwise w.r.t. the anchor – and according to the sorted order.

Let us consider only the simpler alternative since the same argument applies to both. First, let us prove the optimality of the layout of the points which have been moved counter-clockwise (i.e., at the right of p in Fig. 6). Suppose there is an optimal layout which does not respect the increasing order. Let S^* be the sum of the circular edge lengths of this optimal solution. Suppose there are two consecutive points o_{i^*} and o_{j^*} counter-clockwise w.r.t. the anchor point s.t. $a_{i^*} - b_{i^*} > a_{j^*} - b_{j^*} > 0$. Thus, o_{i^*} and o_{j^*} have been moved counter-clockwise by an angle of $inc \times \Theta$ and $(inc + 1) \times \Theta$, respectively. The value inc is the incremental angular difference between adjacent exterior points. The weight of the these two consecutive points in S^* is

$$(b_{i^*} + r^* - a_{i^*}) \times inc \times \Theta + (b_{j^*} + r^* - a_{j^*}) \times (inc + 1) \times \Theta.$$

By permuting the two points, a smaller weight can be obtained. This contradicts the optimality of the solution.

Finally, suppose there is a point $o_{i^*} \in B$ which has been also moved counter-clockwise (a shaded box in Fig. 6). Since $(b_{i^*} - a_{i^*}) > 0$, this point must be closer to the anchor than any other point in A in any optimal layout. Otherwise, by permuting these points, a smaller sum of the circular edge lengths would be obtained. Similar arguments can be used to prove the optimality of the layout of the points which have been moved clockwise (i.e. at the left of p in Fig. 6). □

Based on this characterization of the optimal layouts, a more formal description of this heuristic is presented in Algorithm 2. The running time of this algorithm is in $O(mn + m^2)$.

Figure 7 presents one example showing that the anchor heuristic out performs the naive algorithm, as expected. A better comparison is presented in Sect. 6.3.

6.3 Empirical Comparison

To perform a more thorough comparison of the proposed heuristics, the different algorithms were applied to the same random bipartite graphs, generated as follows. There are ten points which have been fixed on the inner circle. There are n points which have to be laid out on the outer circle. These points have to be connected to the inner points as follow:

- 10% of the exterior nodes have 5 edges.
- 20% of the exterior nodes have 4 edges.

```
1  Find the n anchors p_1, ··· , p_n determined by the interior points (by Lem. 2).
2  for each exterior point o_i do
3      Find the optimal anchor p_{j_i}.
4      Add o_i to the anchor list C_{j_i}.
5  for each anchor list C_{j_i} do
6      Find the optimal position of the |C_{j_i}| points around p_{j_i}.
7      Let A and B be the set of points as defined in Lem. 3
8      for i ← 0 to |A| do
9          for j ← 0 to |B| do
10             Find the optimal solutions with i points of A and j points of B
               which have been moved clockwise (by Lem. 3).
11     Find the optimal solutions among all the solutions in the previous step.
```

Algorithm 2. Anchor heuristic

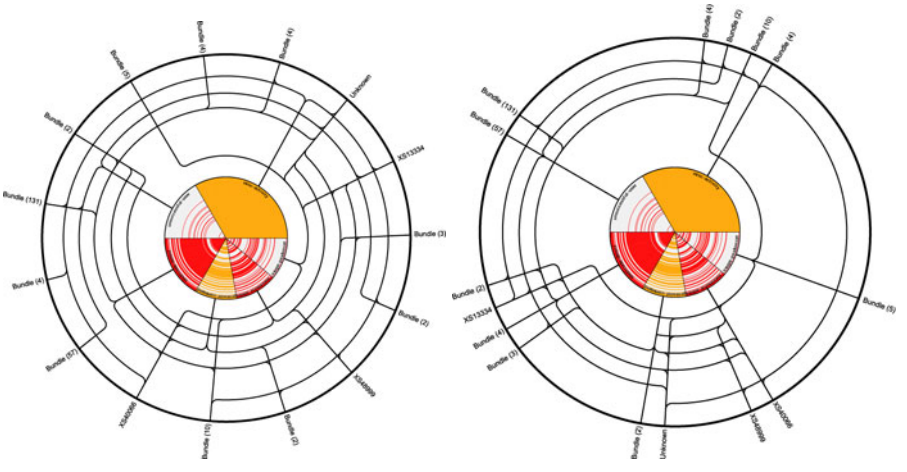

Fig. 7. AlertWheel tool: Comparing the naive algorithm (left) and the anchor heuristic (right). In the former case, the sum of the circular edge lengths is 22036 units. In the latter case, the sum is 17184, or 22% less.

- 30% of the exterior nodes have 3 edges.
- 20% of the exterior nodes have 2 edges.
- 20% of the exterior nodes have only one edge.

For each of these exterior points, their connected neighbors are randomly selected among the ten interior points.

The results of the experiment are presented in Fig. 8. For each number of nodes, 20 bipartite graphs have been generated. The figure shows the average of the total sum of circular edge lengths for each algorithm.

The performance of the algorithms can be compared with a theoretical lower bound. The lower bound is found by finding the optimal position of each exterior

point (as given by Lem. 2) and allowing exterior points to coincide. Thus, this lower bound is unachievable in practice. As expected, the anchor heuristic gives very good results. If the number of competing exterior points for a given anchor is small, each point should be very close to its optimal solution. This should yield a solution that is close to the globally optimal solution.

To conclude, it should be mentioned that the anchor heuristic has a disadvantage for the AlertWheel visualization tool. Because the exterior points are non uniformly distributed on the outer circle, a radial labelling of the nodes has to be used instead of a circular labelling (as in Fig. 5 and 7).

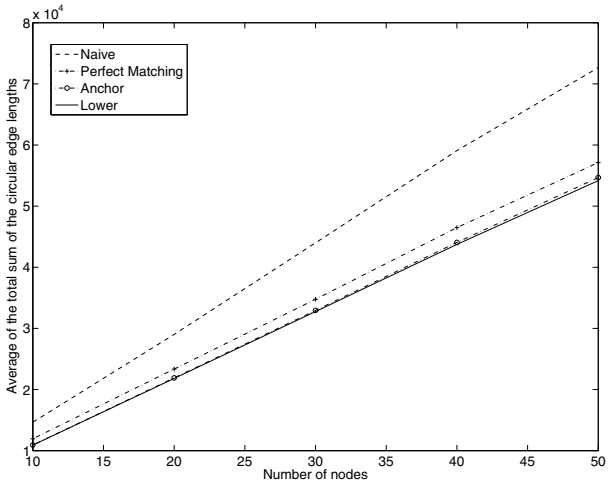

Fig. 8. Comparing the performance of the different algorithms on random bipartite graphs with $n = 10, 20, 10, 40$ and 50 nodes

7 Conclusions

In addition to the user interaction problems that had to be solved during the development of AlertWheel, multiple bipartite graph drawing problems also had to be addressed. In all cases, the naive approaches to laying out the graph without any optimization give poor results. The large quantities of information to deal with have necessitated finding good heuristics to reduce the clutter in the drawing of the radial representation of the bipartite graph representing the observed security alerts of an IDS. One of these heuristics gives very good results which are close the globally optimal solution.

References

1. Abdullah, K., Lee, C., Conti, G., Copeland, J.A., Stasko, J.: IDS RainStorm: visualizing IDS alarms. In: IEEE Workshop on Visualization for Computer Security, pp. 1–10 (2005)
2. Bachmaier, C.: A radial adaptation of the Sugiyama framework for visualizing hierarchical information. IEEE Transactions on Visualization and Computer Graphics 13, 583–594 (2007)
3. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press (2009)
4. Dumas, M., Robert, J.M., McGuffin, M.J., Willig, M.C.: AlertWheel: Radial bipartite graph visualization applied to intrusion detection system alerts (submitted for publication)
5. Foresti, S., Agutter, J., Livnat, Y., Moon, S., Erbacher, R.: Visual correlation of network alerts. IEEE Comput. Graph. Appl. 26, 48–59 (2006)
6. Gansner, E., Koren, Y.: Improved Circular Layouts. In: Kaufmann, M., Wagner, D. (eds.) GD 2006. LNCS, vol. 4372, pp. 386–398. Springer, Heidelberg (2007)
7. Northcutt, S., Novak, J.: Network Intrusion Detection, 3rd edn. New Riders, Indianapolis (2002)
8. Pupyrev, S., Nachmanson, L., Kaufmann, M.: Improving Layered Graph Layouts with Edge Bundling. In: Brandes, U., Cornelsen, S. (eds.) GD 2010. LNCS, vol. 6502, pp. 329–340. Springer, Heidelberg (2011)
9. Purchase, H.C.: Which Aesthetic has the Greatest Effect on Human Understanding? In: Di Battista, G. (ed.) GD 1997. LNCS, vol. 1353, pp. 248–261. Springer, Heidelberg (1997)
10. Tamassia, R., Palazzi, B., Papamanthou, C.: Graph Drawing for Security Visualization. In: Tollis, I.G., Patrignani, M. (eds.) GD 2008. LNCS, vol. 5417, pp. 2–13. Springer, Heidelberg (2009)
11. Zheng, L., Song, L., Eades, P.: Crossing minimization problems of drawing bipartite graphs in two clusters. In: Asia-Pacific Symposium on Information Visualisation, pp. 33–37 (2005)