Network drawing with geographical constraints on vertices

Manuel Abellanas * Andrés Aiello † Gregorio Hernández Peñalver ‡
Rodrigo I. Silveira §

Abstract

In this paper we concentrate on an unexplored graph drawing problem: the one arising when each vertex of the graph represents a geographical region. For this problem we first define new aesthetic criteria. Secondly, several force-directed algorithms for finding layouts that satisfy those specific criteria are proposed, which successfully obtain much better drawings for this problem than the previous algorithms.

1 Introduction

Networks, or graphs, arise in a natural way in many applications, together with the need to be drawn. Except for very small instances, drawing a network by hand becomes a very complex task, which must be performed by automatic tools. The field of graph drawing is concerned with finding algorithms to draw graphs in an aesthetically pleasant way, based upon a certain number of aesthetic criteria that define what a good drawing, or layout, of a graph should be. Although many powerful graph drawing algorithms have been designed, when faced with the additional requirement that vertices represent geographical regions – and that this must be reflected in the drawing – these algorithms turn ineffective.

This problem can be found in many areas, such as in the visualization of data networks. An example can be found in Figure 1, showing the IRIS network, that connects universities and research centers in Spain. The figure shows the data links between the Spanish Autonomous Communities. In this example each vertex has associated with it some geographical information (the part of Spain it represents), and the drawing of the graph must be consistent with that. This adds to the usual aesthetic criteria the constraint that each vertex must be drawn inside its associated community.

From the graph drawing point of view, we are facing a standard drawing problem with two additional constraints: vertices must be drawn inside their regions and each vertex must represent its region in the best possible way (we will call this kind of constraints *geographical constraints*). More formally:

Problem statement. Given a graph $G = (V, E)$ and a set of geographical regions[1] R, where $r_v \in R$ is the non-empty region of the plane *represented* by vertex $v \in V$, find a drawing of G in two

*Departamento de Matemática Aplicada, Facultad de Informática, Universidad Politécnica de Madrid, Spain. mabellanas@fi.upm.es

†Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina. aaiello@dc.uba.ar

‡Departamento de Matemática Aplicada, Facultad de Informática, Universidad Politécnica de Madrid, Spain. gregorio@fi.upm.es

§Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina. rsilveir@dc.uba.ar

Partially supported by MCyT project TIC2003-08933-C02-01.

[1] Along this work we will suppose the regions are disjoint polygons.

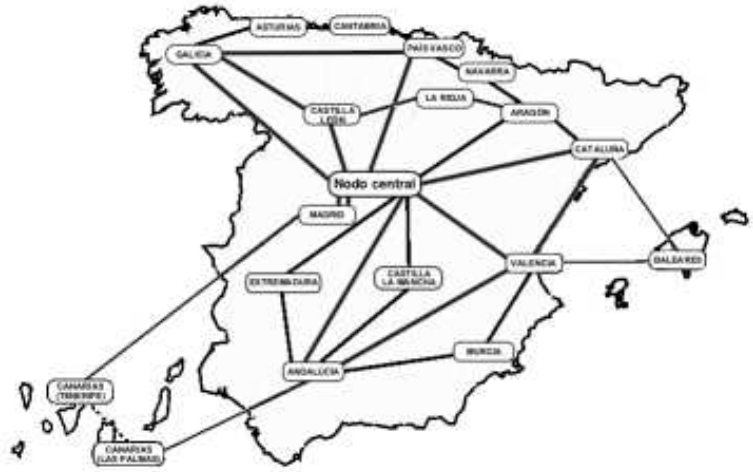

Figure 1: A drawing of the IRIS network, extracted from `www.rediris.es`.

dimensions that is aesthetically pleasant, and where $p_v \in r_v$ for all vertex $v \in V$ (p_v is the position of v in the drawing).

We emphasize that each vertex must *represent* its region to stress that not only must vertices be drawn inside their regions, but also must represent them. Edges in the drawing should look like edges between the regions, instead of between the vertices.

2 Related work

The field of graph drawing is concerned with finding efficient algorithms to visualize graphs, in the best possible way. This is usually defined through the so-called "aesthetics criteria", which specify what properties a good layout of a graph must have (for example, it should not have too many edge crossings). During the last years lots of graph drawing algorithms have been published [3] [8]: layered, orthogonal, for planar graphs, etc. One of the most important families of graph drawing algorithms is the one corresponding to force-directed methods. They make use of physical analogies to draw the graph, and are among the most commonly used algorithms because they are intuitive, easy to program, they yield good results and are very flexible. The algorithms proposed in this paper belong to this class.

In our particular case, the standard graph drawing problem is extended with the aforementioned constraints. We have not found in the literature any other publication dealing with this specific kind of constraints, and that analyzes this visualization problem as a whole: vertices that *represent* regions (that, as we will see, is much different from vertices *constrained* to their regions).

There is a lot of previous research on algorithms that consider different kinds of constraints on the positions of vertices. The most common and simple are relative or absolute constraints [7] [9], but more complex, linear [6] or even non-linear constraints [5] on the positions of vertices have also been studied. The latter make use of specific methods for dealing with constraints, such as general constraint solvers. The constraints we deal with in this work are not relative or absolute positions. The particular case of convex regions could be tackled with an algorithm like the one of [6], but concave regions could only be included in general constraints algorithms like the one of [5].

In any case, using these algorithms involves using tools that are too general for a very specific problem. This is why it is possible to take a much simpler approach and adapt an iterative force-directed algorithm to constrain the movement of vertices to the interior of their regions. However, as it will be shown in the next section, this is not enough to yield good results, due to the fact that each vertex representing a region implies aesthetic considerations that go beyond constraining its position to an area of the plane.

3 Aesthetic criteria

The stated problem presents two main aspects. On the one hand, the position of each vertex needs to be limited to its region. On the other hand, like in any graph drawing problem, we look for an aesthetically pleasant layout of the graph. This means that the information in it should be conveyed in the best possible way. When dealing with graphs whose vertices represent geographical regions, this second aspect turns out to be specially important, even though at first glance it might seem that the usual graph drawing criteria are enough to obtain good quality results. In this section we will show that this is not the case, and that new aesthetic criteria need to be defined for this new problem.

3.1 A first approach

At first glance it might seem like the only necessary change to deal with this problem is to solve the first aspect, that is, to constrain each vertex to its region. This can be implemented in a very simple way. If we use a force-directed algorithm based on the spring embedder [4], it is enough check on each step that no vertex is positioned outside its region. Even though this would be enough to get a feasible drawing, it is not hard to realize that an algorithm that yields good results for the standard problem, might not still do it for the constrained one. The drawing on the left of Figure 2 shows an example. A simple graph was drawn with a variant of the spring embedder, constraining the movements of each vertex on each step. If we did not have the geographical regions, the layout would be pretty good. But with vertices representing regions, it is evident that the layout is not good at all. It is not clear how the regions are connected, that is precisely what the drawing must communicate. A much better drawing of the same graph is shown on the picture on the right.

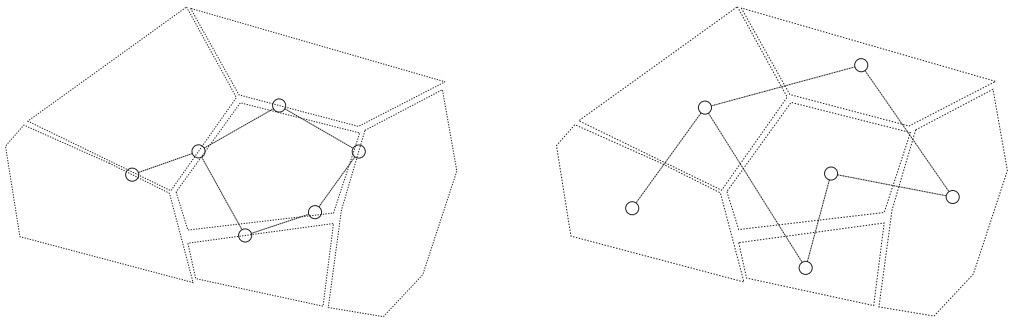

Figure 2: Left: poor layout obtained by only constraining the position of each vertex to its region. Right: a layout that shows in a better way the connections between the regions of the same graph.

This example shows that **layouts produced considering only the constraints on the positions of the vertices are not necessarily good layouts for our problem**. This has two important implications. Firstly, existing graph drawing algorithms cannot be applied blindly to this new problem. Secondly, it must be redefined what a good layout, for our problem, is. In this section we will study the aesthetic criteria that define, in our problem, what a good layout is.

3.2 Aesthetic criteria for this problem

We will begin by reviewing the aesthetic criteria that are valid in this problem of graph drawing with geographical constraints. In a standard graph drawing scenario there is a large number of aesthetic criteria (minimizing the number of edge crossings, maximizing the angle between adjacent edges or edges that cross, obtaining uniform edge length, just to name a few). See [3], [8] for a complete review. However, when vertices represent geographical regions, many of those criteria stop being valid and new criteria arise.

For example, typical force-directed algorithms strive for uniform vertex distribution and uniform edge length. However, in our problem none of these criteria is valid. This explains why previous algorithms do not yield good results.

The special requirements of this problem give rise to new aesthetic criteria. The tests run to determine what were the previous criteria that yielded the best results showed that crossing minimization and appropriate edge lengths[2] were among the most influential ones. Nevertheless, we noticed they were not enough to obtain good results. An example is shown in Figure 3. Even though the edge length is very good (close to uniform) and there are no crossings, the drawing is not successful in transmitting the connections between the regions in a clear way.

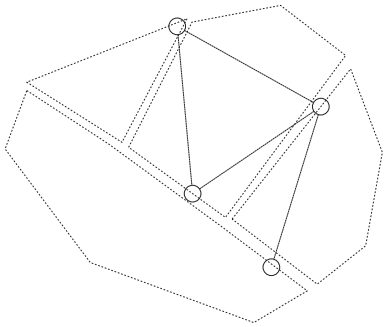

Figure 3: New aesthetic criteria for this problem are needed.

Results like the previous one motivated the search for new criteria that allowed the algorithms to yield better layouts. The two criteria that were found to be most important are presented next.

The first one is that **vertices should be placed near the center of its regions**. Vertices must represent their regions, and this implies that the positions where they are located should allow to easily identify which region is represented by each vertex. In general, the center of the region is the point that best represents it. In addition to this, vertices close to the borders of their regions add confusion over what the region the vertex represents is (in relation to this we also propose the next criterion). Another good property of this criterion is that in many cases, when the distance between the borders and the center of the regions is uniform, it allows to produce drawings with uniform edge lengths, a property that is also considered beneficial.

The second new criterion states that **vertices should not be placed near the borders of their regions**, because this makes it difficult to identify the region each vertex belongs to, and sometimes edges end up drawn parallel to a border of the region, making it even harder to understand the layout. Although the previous criterion might imply this one, we think this is a criterion itself that should be considered separately.

4 Proposed algorithms

In this work we propose algorithms based on two well-known force-directed algorithms: the DH algorithm of Davidson and Harel [2] and the spring embedder (SE) of Eades [4]. Their flexibility allows us to include in a straightforward way the new aesthetic criteria presented in the previous section. To constrain each vertex to its region, on each step of the algorithm its new position is projected to the closest point in its region. In this section we describe the changes introduced to the DH and SE algorithms.

4.1 Ideal edge length

The first criterion studied was the ideal or natural length of the edges. Clearly, this ideal length should take the regions into account. In the standard graph drawing problem the ideal length is usually constant, but, to begin with, in our problem the geographical constraints impose minimum and maximum lengths for each edge, that can vary arbitrarily. Several test were run to help devise what the ideal length should be. We found that each edge needed an individual length, which considered

[2]This refers to the idea that vertices with short graph-theoretic distances should be drawn close to each other.

the distance between the two regions involved. The first length chosen for each edge (u, v) was the distance between their centers: $K_{uv} = \|c_u - c_v\|_2$.

In our implementation we decided to define the center of a region as its center of mass or centroid. We found this adequate because it is independent of the density of the vertices of the polygon that defines the region and because it can be computed efficiently.

The introduction of this variable ideal length quickly improved the results, although it was noticed that vertices were attracted too strongly to the centers. Another problem observed was that in exterior regions – the ones that share a border with the unused workspace – sometimes this length was too large. For this reason we adopted a *compromise solution* that kept the vertices close to the center but at the same time allowed for some more freedom in their positions. This ideal length was defined as follows:

$$K_{uv} = \lambda \|c_u - c_v\|_2 + (1 - \lambda) \|r_u - r_v\|_2 \tag{1}$$

where $\lambda \in (0, 1)$ and $\|r_u - r_v\|_2$ is the minimum distance between the two regions involved. In our test we used most of the time $\lambda = 0.5$, that is, the average between the distance between the centers of the regions and their minimum distance.

The results obtained with this new length were much better than with the previous one, resulting in many cases in a very good topology for the layout. A simple example is shown in Figure 4. Based on the tests we ran on different input graphs, we concluded that this compromise distance yielded the best results.

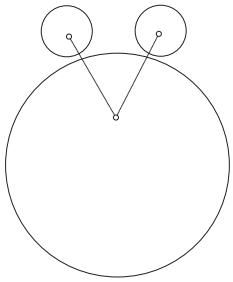

Figure 4: Layout obtained using the *compromise distance.*

4.2 Forces towards the center

The next criterion to be considered was that vertices should be placed near the center of their regions. For this purpose we decided to add to the spring embedder a centripetal attractive force that prevented vertices from being placed too far from the center. This way, the forces involved in the algorithm were the usual attractive and repulsive ones of the spring embedder, in addition to the new centripetal forces.

The expression of the **centripetal forces**, defined for each vertex $v \in V$ is:

$$f_c(v) = C_1 \log(\frac{d_{c_v v}}{\varepsilon})\overrightarrow{p_v c_v} \tag{2}$$

where $d_{c_v v}$ is the distance between v and the center of its region, c_v, and $\varepsilon < 1$ and C_1 are constants. This can be thought as adding a fictitious fixed vertex at the center of each region, with one edge that connects it to the vertex representing the region. The ideal length of this fictitious edge is a very small ε, so as to force the other vertex to move towards the center. A logarithmic force was used in order to make this centripetal forces of the same kind as the attractive ones.

The use of this forces improved the results in a significant way. An example is shown in Figure 5 where a simple graph was drawn with and without centripetal forces.

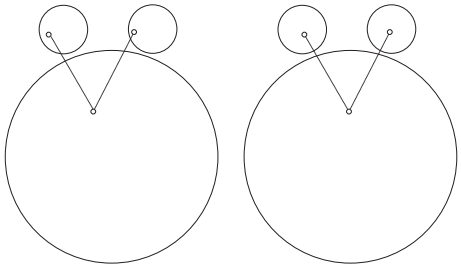

Figure 5: Improvement caused by the addition of centripetal forces. The same graph was laid out without (left) and with (right) centripetal forces.

Although the improvement caused by the addition of these forces was significant, managing to attract the vertices towards the center of their regions successfully, we introduced another change to make the centripetal forces stronger when the vertices where located close to the borders, and in this way speed up the convergence time of the algorithms. We tried several changes, and what gave the best results was simply calculating the magnitude of the forces as the square of the expression 2. This change also required new values for the constant C_1, that empirically was set to a number between 0.5 and 0.8.

Besides adding this centripetal forces to SE, we also added an equivalent potential to the DH algorithm. The energy function of the algorithms was extended with a potential that penalizes the distance between each vertex and the center of its region. It was defined for every vertex $v \in V$ as $U_c(v) = d^3_{c_v v}$.

Notice that the complexity of both algorithms, SE and DH, remains the same after the addition of the centripetal forces.

4.3 Minimize the number of edge crossings

The criterion that states that the number of edge crossings should be minimized is one of the most difficult criteria to implement, and only a few graph drawing algorithms for general graphs consider it in a explicit way. Since our experimental tests showed that minimizing edge crossings still is a very important criterion for our graph drawing problem, we decided to attack this problem using an algorithm based on DH.

The DH algorithm already contains a discrete potential that penalizes edge crossings. However, besides being discrete, it does not guide the optimization process towards solutions with fewer crossings. For this reason we decided to add a new potential that has two important features: it is continuous, which helps the optimization method (Simulated Annealing) to converge to a minimum, and secondly, it helps to eliminate certain kinds of crossings, guiding the search to better solutions. For reasons of brevity we will not give any details about this potential here. We refer the interested reader to [1].

4.4 Avoid vertices close to borders

In order to add this new criterion to the algorithms we added a potential to DH that adds a penalty when a vertex is too close to a border of its region. We looked for a function to repel strongly the vertex when it gets close to the border, but whose strength quickly decreases when the vertex gets away. After experimenting with several functions, we chose the following one (defined for every $v \in V$):

$$U(v) = c_{distToBorder} e^{(-2 * \beta * distToBorder)} \tag{3}$$

This functions approaches $c_{distToBorder}$ when the distance between the vertex and the border $distToBorder$ approaches zero, while the function approaches zero when the distance grows. The constant β is an adjustment factor to control how fast the potential approaches zero.

Using this potential the algorithm was able to strongly repel the vertices from the border, but at the same time give them enough freedom to move once they were away. Empirically we set $\beta = 0.2$ and $c_{distBorde} \approx 10^5$.

4.5 Implementation: SER^2 y DHR^2

Based on the ideas presented so far we implemented two algorithms, named SER^2 and DHR^2, (R^2 stands for *Restricted to Regions*). The first one is the adaptation of the spring embedder while the second one is the adaptation of the DH algorithm. These algorithms take into account all the considerations discussed in the previous sections, including the potentials to reduce the number of crossings (not described here). The algorithms were implemented as part of a Java application that was created extending the existing graph drawing public license tool VGJ[3].

5 Results

In this section we present two examples where our algorithms have been used to layout two graphs with geographical constraints. The first one shows a graph connecting the provinces of Argentina (Figure 6), while the second one corresponds to the already mentioned IRIS network, from Spain (Figure 7). For each graph we present three layouts: the initial (random) layout used, the one generated with SER^2 and the one generated with DHR^2.

Figure 6: Example of a map of Argentina. Initial layout (left), layout produced by SE-R^2 (middle) and layout produced by DH-R^2 (right).

It is very difficult to analyze and compare this kind of results, since there is no *natural way* to draw these graphs. In any case, the resulting layouts are very successful in showing the connections between the regions. All of them improve the initial layouts. In the case of the Argentinean map, the number of crossings in the layout by SER^2 was reduced and, in general, both manage to improve the initial one in a significant way. Both drawings of the IRIS network of Spain are also aesthetically very good, with an aesthetic quality similar to the original drawing from the IRIS website (Figure 1).

[3]Available for download at
`http://www.eng.auburn.edu/department/cse/research/graph_drawing/graph_drawing.html`

Figure 7: Example of a map of Spain (IRIS network). Initial layout (left), layout produced by SE-R^2 (middle) and layout produced by DH-R^2 (right).

6 Conclusion and Future Work

We have presented effective algorithmic solutions for the graph drawing problem that arises when each vertex represents a geographical region. For this previously unexplored problem the existing aesthetic criteria are analyzed, studying which of them are still valid and which are not, and two important new criteria that help define – in this context – what a good layout is, are proposed. In addition, two algorithms are presented, both extensions of well-known force-directed methods, that successfully implement the proposed aesthetic criteria, obtaining drawings with the desired properties that improve the layouts that could be obtained with the existing graph drawing algorithms.

While our algorithms represent a major improvement in the quality of the results that can be obtained for this graph drawing problem, there are many directions of research that deserve to be explored. The algorithms proposed in this paper are not able to handle large graphs (thousands of vertices) efficiently. It would be interesting to study how to take advantage of the information available about the regions to be able to draw larger graphs in a faster way. It is also of interest the study of specific classes of regions, such as concave regions or regions that induce a partition of the plane.

References

[1] A. Aiello and R. Silveira. Trazado de grafos mediante métodos dirigidos por fuerzas: revisión del estado del arte y presentación de algoritmos para grafos donde los vértices son regiones geográficas. Master's thesis, Universidad de Buenos Aires, Argentina, 2004.

[2] R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15(4):301–331, 1996.

[3] G. DiBattista, P. Eades, R. Tamassia, and I. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, first edition, 1999.

[4] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, (42):149–160, 1984.

[5] T. Hansen, K. Marriott, B. Meyer, and P. Stuckey. Flexible graph layout for the web. *Journal of Visual Languages and Computing*, 13(1):35–60, 2002.

[6] W. He and K Marriott. Constrained graph layout. *Constraints*, 3(4):289–314, 1998.

[7] T. Kamps, J. Kleinz, and J. Read. Constraint-based spring-model algorithm for graph layout. In *Proceedings of Graph Drawing '95*, pages 349–360. Springer-Verlag, 1995.

[8] M. Kaufmann and D. Wagner, editors. *Drawing graphs: methods and models*, volume 2025 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[9] X. Wang and I. Miyamoto. *Generating customized layouts*, pages 504–515. Springer-Verlag, 1995.