Vis Comput (2012) 28:1015–1026
DOI 10.1007/s00371-012-0727-7

ORIGINAL ARTICLE

Generating optimal drawings of physically realizable symbol maps with integer programming

Guilherme Kunigami · Pedro J. de Rezende · Cid C. de Souza · Tallys Yunes

Published online: 26 May 2012

Abstract Proportional symbol maps are a tool often used by cartographers and geoscience professionals to visualize geopositioned data associated with events and demographic statistics, such as earthquakes and population counts. Symbols are placed at specific locations on a map, and their areas are scaled to become proportional to the magnitudes of the data points they represent. We focus specifically on creating physically realizable drawings of symbols—opaque disks, in our case—by maximizing two quality metrics: the total and the minimum length of their visible borders. As these two maximization problems have been proven to be NP-hard, we provide integer programming formulations for their solution, along with decomposition techniques designed to decrease the size of input instances. Our computational experiments, which use real-life data sets, demonstrate the effectiveness of our approach and provide, for the first time, a number of optimal solutions to previously studied instances of this problem.

Keywords Visualization · Cartography · Computational geometry · Integer linear programming

G. Kunigami (✉) · P.J. de Rezende · C.C. de Souza
Institute of Computing, University of Campinas, Campinas, Brazil
e-mail: kunigami@gmail.com

P.J. de Rezende
e-mail: rezende@ic.unicamp.br

C.C. de Souza
e-mail: cid@ic.unicamp.br

T. Yunes
Management Science Department, University of Miami, Coral Gables, FL, USA
e-mail: tallys@miami.edu

1 Introduction

Proportional symbol maps are a cartographic tool employed in the visualization of geopositioned data or events associated with specific locations. In these maps, symbols are placed over the points that correspond to the positions where data were gathered or events occurred, and the area of these symbols are made proportional to the magnitude of the phenomenon they represent. Commonly represented data include earthquakes (with location and intensity), and demographic statistics. While symbol shapes vary according to each application, disks are often a very intuitive form of conveying information on the magnitude of events [4]. Hence, we restrict ourselves to the placement of opaque circles. Clearly, due to the proximity of the disks and their sizes, overlapping may occur, as shown in Fig. 1.

Depending on the scaling factor applied to the symbols, the amount of overlapping can differ greatly. Although the general rule for choosing the representation scale, as stated by Slocum et al. [9] "neither too full nor too empty," is rather subjective, it is expected that any visually pleasing map will contain at least some overlap of symbols. Depending on the (partial) order in which the disks are organized, different portions of the symbols will be visible. The question we address here is how to arrange a given set of disks so that the final map reveals the best possible visual information.

The first paper to address this problem algorithmically was Cabello et al. [1]. They introduced two metrics to quantify the quality of a drawing and also two possible drawings of disks, leading to four very interesting and related problems. We now briefly describe them.

Employing the same definitions and notations as in [1], let S be a set of n disks in the plane. An arrangement $\mathcal{A}$ of the boundaries of these disks partitions the plane into connected regions. A vertex of $\mathcal{A}$ is the intersection point of

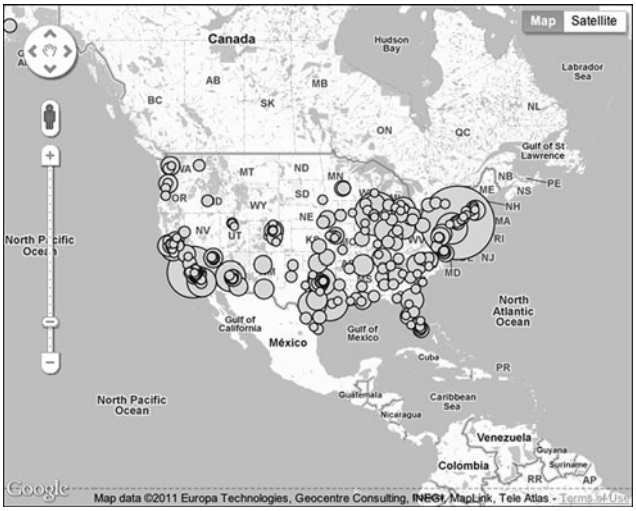


Fig. 1 A screen shot from an interface that generates proportional symbol maps from a data set

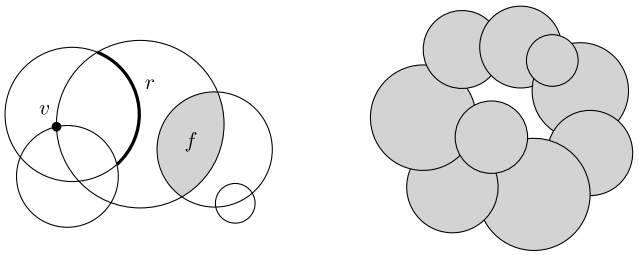


Fig. 2 Arrangement with vertex v, arc r, and face f (*left*); a physically realizable drawing with interleaving disks (*right*)

two or more boundaries. An arc of $\mathcal{A}$ is a maximal connected portion of the boundary that connects two vertices and contains no vertex in its interior. A face of $\mathcal{A}$ is a maximal connected region bounded by arcs that does not contain any vertices or arcs in its interior. A drawing $\mathcal{D}$ of S is a subset of arcs and vertices of $\mathcal{A}$ drawn on top of the filled interiors of disks in S. Figure 2 shows an arrangement and a drawing.

A physically realizable drawing can be constructed from whole symbols cut out from sheets of paper. Disks can be interleaved as in Fig. 2 (right), provided that physical restrictions are observed. The drawing in Fig. 3 cannot be created without cutting the disks.

Consider the arrangement of four disks in Fig. 4(i). If the disks are of different colors, it is pretty clear that in any physically realizable drawing each face of the arrangement contained in at least one disk will have a unique color. By not allowing the disks to be cut or folded, we have that given two intersecting disks d_i and d_j, either d_i is over d_j (denoted "$d_i > d_j$"), or vice-versa. Thus, the color that is seen on a face f corresponds to that of the disk that is placed over all other disks that contain f. The iterated removal of the topmost disk, and the corresponding change in the

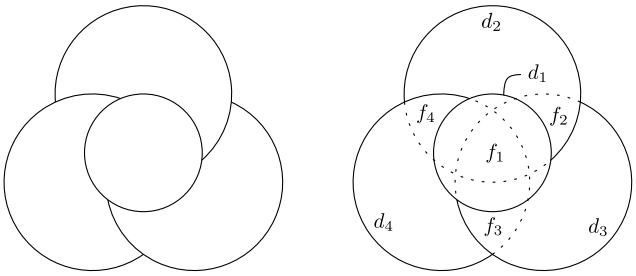


Fig. 3 A drawing that is not physically realizable and the underlying arrangement. If we remove the topmost disk, how can the remaining ones be arranged?

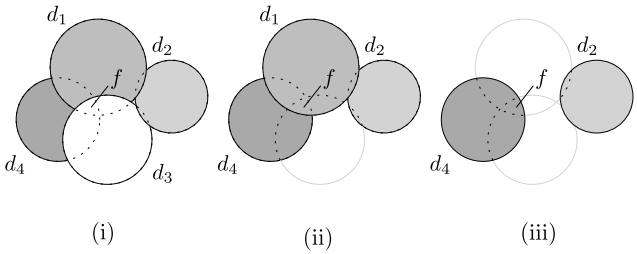


Fig. 4 A physically realizable drawing of fours disks. The region of face f from the arrangement that is initially seen in (**i**) belongs to d_3. After removing d_3, it belongs to d_1 in (**ii**), and after removing d_1, it belongs to d_4 in (**iii**)

color of f, induces a total ordering of the disks containing f.

This ordering alone, however, is not enough to define a physically realizable drawing. For example, although the drawing in Fig. 3 has an order-inducing sequence of colors for each face, it cannot be physically constructed. To see why, notice that face f_1 induces a total order between d_1, d_2, d_3, and d_4, but any such order will conflict with the orders $d_2 > d_3$, $d_3 > d_4$, and $d_4 > d_2$ induced by faces f_2, f_3, and f_4, respectively. In other words, although physically realizable drawings do not require a total order, multiple partial orderings of the disks engendered by the faces must not contradict each other.

When the requirement of total order among the disks of S is added, that is, all cyclic orders become forbidden, we have a special case of a physically realizable drawing known as a *stacking drawing*.

An arc r from the arrangement will belong to a drawing (i.e., it will be visible) when it is not covered by any disk. That is, the disk having r in its border must be above all disks that contain r in their interior.

According to Cabello et al. [1], good drawings should enable the viewer to see at least some part of every disk and to gauge their sizes as correctly as possible. As supporting evidence for this argument, consider Fig. 5, in which we can determine neither the center coordinates nor the radius of the hidden disk. This led to the definition of two metrics

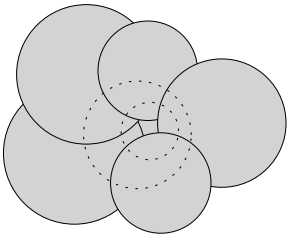


Fig. 5 An example of a disk whose border is completely covered. Its center and radius cannot be determined

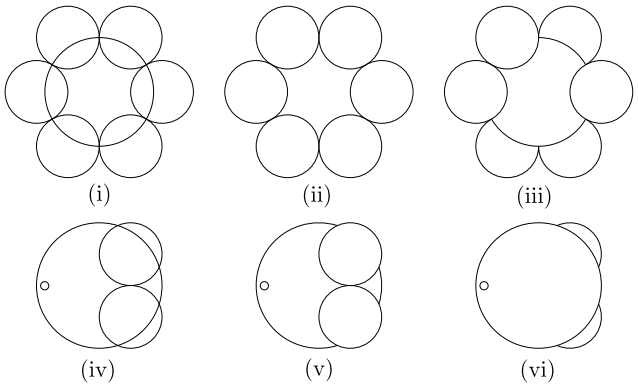


Fig. 6 (**ii**), (**iii**) are solutions for (**i**) optimizing Max–Total and Max–Min metrics, respectively; analogously, (**v**) and (**vi**) optimize Max–Total and Max–Min metrics for instance (**iv**)

used to quantify the quality of a drawing $\mathcal{D}$ of a set S of disks: the minimum visible boundary and the total visible boundary of all disks in S. If b_i is the total length of the visible boundary of disk $i \in S$ in drawing $\mathcal{D}$, we can state four different optimization problems: maximizing $\min\{b_i | i \in S\}$ (Max–Min) or maximizing $\sum_{i \in S} b_i$ (Max–Total), either for physically realizable drawings or for stacking drawings. At this point, we should say that despite the attempts to capture the essence of good drawings, these two metrics proposed earlier in the literature may not be capable to produce good drawings in all cases. As an example, consider the instance given in Fig. 6(i). The optimal solutions for the Max–Total and the Max–Min metrics are shown in (ii) and (iii), respectively. One can see that in (ii) the bottom disk is totally covered, as it happens in Fig. 5. Likewise, for the instance in Fig. 6(iv), optimal solutions for Max–Total and Max–Min are depicted in (v) and (vi), respectively. Note that the position of the tiny disk determines the Max–Min value, whereas the order of the remaining disks is irrelevant for this objective function. As a result, there may be optimal Max–Min solutions for this instance that do not necessarily represent good drawings, as is the case in (vi). Notwithstanding these extreme cases, these metrics have been shown to produce high-quality drawings in practice.

For the remainder of this paper, we refer to the physically realizable drawing problem as PRDP.

1.1 Related work

Cabello et al. [1] show that PRDP is NP-hard in both the Max–Min and Max–Total versions. They also present an $O(n^2 \log n)$ algorithm for the Max–Min version restricted to stacking drawings, whereas the complexity of the Max–Total stacking drawing problem remains open.

An integer linear programming (ILP) formulation for the Max–Total stacking drawing problem together with a theoretical study of that formulation is presented in [8]. Since the difficulty of finding optimal solutions grows with the cardinality of S, two techniques for decomposing a given instance into smaller ones are also studied in [8]. In [6], a tighter ILP formulation for the same problem was presented which turned out to be much more effective in solving the same set of instances.

1.2 Our contributions

This paper extends the results in [7], where we presented the first exact algorithm for the Max–Total version of the PRDP, by providing an ILP formulation and a theoretical study of the resulting model. Moreover, we describe a novel decomposition technique that can be successfully used, along with previously introduced ones, to significantly speed up the solution of large instances. We have been able to find optimal solutions for several data sets for which only approximate solutions were previously known, and also for a number of very hard instances derived from demographic statistics. Furthermore, we also describe the first exact algorithm for the Max–Min variant of the PRDP: by introducing a real-valued variable, we were able to convert the previous model into a mixed integer linear programming (MILP) formulation for the Max–Min version. We also draw on results obtained for the Max–Total PRDP in [7] to derive a new algorithm for the Max–Total Stacking Drawing which is significantly faster than the one presented in [6]. To our knowledge, we are the first to provide provably optimal solutions for both the Max–Total and Max–Min versions of the PRDP, and the first to use integer programming to tackle these problems.

1.3 Organization

In Sect. 2, we provide a brief background on ILP and MILP. In Sect. 3, we describe the optimization models for both stacking and physically realizable drawings, including extra inequalities that can be added to strengthen these models. In Sect. 4, we describe the decomposition techniques presented in [8] that remain applicable here, as well as a new one developed specifically for the Max–Total PRDP. Details of our implementation and experiments appear in Sects. 5 and 6, respectively. Section 7 contains an analysis of our results, followed by our final conclusions in Sect. 8.

2 Integer linear programming background

We now provide a brief overview of basic integer programming concepts. Wolsey's book [10] is an excellent reference for further reading on this subject.

Given a set of variables $x \in R^n$, a *linear programming* (LP) problem consists of minimizing a linear function $c^T x$ subject to a set of linear constraints $Ax \leq b$, where $c \in R^n$, $A \in R^{m \times n}$, and $b \in R^m$. If, in addition, all (some) of the x variables are restricted to be integer, we have an *integer (mixed-integer) linear programming* problem (ILP/MILP). Although LP problems can be solved in polynomial time, solving an ILP/MILP problem is NP-Hard in general [10].

Let P be the set of integer points satisfying the constraints of an ILP model and consider the convex hull, conv(P), of P. Note that an optimal solution to the ILP is to be found on a vertex of conv(P). Therefore, if we could find a polynomial description of conv(P), we might ignore the integrality constraint of the variables and solve the model in time polynomial on the size of the constraints that describe conv(P).

In general, the number of inequalities to define this convex hull is exponential in the size of the input. To circumvent this obstacle in practice, one uses a *valid formulation* which is any formulation (i.e., system of inequalities) that contains all points in P and no other integer points. The corresponding model with variables restricted to integers is then solved with a branch-and-bound (B&B) algorithm. These algorithms have theoretical exponential running times, but tend to behave well in practice. Clearly, there are infinitely many valid formulations for a given set P, and those that are closer to the convex hull lead to better running times.

With this in mind, one seeks concise families of valid inequalities for the problem. Since there may exist an exponential number of them, some selection process needs to come into play. For this, we use *cutting plane algorithms* which consist of solving in polynomial time a linear relaxation of the model (that is, a model without the integrality constraint on the variables) to find an optimal solution s^*. If this solution is integer, the problem is solved. Otherwise, we solve the *separation problem* which, given a family of valid inequalities, consists of (i) finding one that cuts off the optimal solution s^*, and (ii) adding it to the formulation. This procedure is repeated a given number of times or until no such inequalities can be found. The resulting formulation is stronger and typically faster to solve.

The best cutting planes are those that define *facets* of the polyhedron, that is, those which have dimension one less than the convex hull of feasible integer solutions. Therefore, when finding a new family of inequalities for a problem it is important to prove whether they are facet-defining.

3 Integer linear programming formulations

We first describe the Max–Total stacking drawing model from [6] because our model is related to it.

3.1 The Max–Total stacking drawing problem

We need the following data, which can be calculated in polynomial time given the set of disks S as input:

- $R \equiv$ set of all arcs of the arrangement;
- $\ell_r \equiv$ length of arc $r \in R$;
- $d_r \equiv$ disk that contains arc r in its border;
- $S_r^I \equiv$ set of disks that contain arc r in their interior.

There are two sets of variables. For each arc $r \in R$, the binary variable x_r equals 1 if arc r is visible, and equals 0 otherwise. The Max–Total problem maximizes

$$\sum_{r \in R} \ell_r x_r \tag{1}$$

For each pair of disks $i, j \in S$, we define the binary variable w_{ij} which is equal to 1 if disk i is above disk j, and equal to 0 otherwise. The constraints are given by:

$$w_{ij} + w_{ji} \leq 1, \quad \forall i,\ j \in S, i < j \tag{2}$$

$$x_r \leq w_{d_r j}, \quad \forall r \in R,\ j \in S_r^I \tag{3}$$

$$w_{ij} + w_{jk} - w_{ik} \leq 1, \quad \forall i \neq j \neq k \neq i \in S \tag{4}$$

$$x_r \in \{0, 1\}, \quad \forall r \in R \tag{5}$$

$$w_{ij} \in \{0, 1\}, \quad \forall i, j \in S, i \neq j. \tag{6}$$

Constraints (2) and (4) ensure that the w_{ij} variables impose a partial order on the disks (see also Proposition 1). Constraint (3) says that if arc r is visible, its disk d_r has to be above all other disks that contain r in their interior.

In [8], additional inequalities are used to strengthen the models. One of those inequalities is based on the fact that some arcs may not be simultaneously visible.

We define a graph $G_I = (V, E)$, with a vertex $v(r) \in V$ corresponding to arc $r \in R$ and an edge $(v(r_1), v(r_2))$ if d_{r_1} contains r_2 and d_{r_2} contains r_1. Two arcs whose vertices are adjacent in G_I cannot both be visible. We can extend this observation to cliques. Given a maximal clique K in G_I, $R(K)$ is the set of arcs with corresponding vertices in K. The following is a valid constraint:

$$\sum_{r \in R(K)} x_r \leq 1, \quad \forall K \in G_I \tag{7}$$

We define a vertex of the arrangement to be non-degenerated if it is formed by the intersection of exactly two circumferences. Given one such vertex v, its neighborhood consists of four incident arcs as in Fig. 7(i). From all

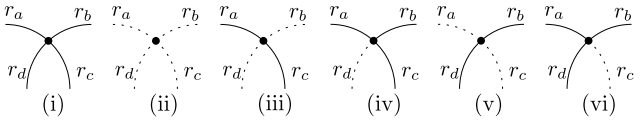


Fig. 7 Arcs incident to a non-degenerated vertex v (*the black dot*) in **(i)**, and feasible configurations in **(ii)**–**(vi)**

16 possible configurations of visible arcs, only five are feasible. They are shown in Fig. 7(ii)–(vi). Together with (7), valid constraints introduced in [8] avoid all infeasible cases. Referring to r_1, r_2, r_3, and r_4 as in Fig. 7(i), those constraints are:

$$x_{r_1} \geq x_{r_3} \tag{8}$$

$$x_{r_2} \geq x_{r_4} \tag{9}$$

$$x_{r_3} + x_{r_4} \geq x_{r_1} \tag{10}$$

$$x_{r_3} + x_{r_4} \geq x_{r_2} \tag{11}$$

3.2 The Max–Total PRDP

In this section, we will show that an ILP formulation corresponding to a given subset of constraints from the previous model is valid for the Max–Total PRDP. Let F be the set of faces from the arrangement. Given $f \in F$, let S_f be the set of disks that contain face f.

Our model is similar to the previous one except that constraint (4) is replaced by the following:

$$w_{ij} + w_{jk} - w_{ik} \leq 1, \quad \forall f \in F,\ i, j, k \in S_f. \tag{12}$$

Intuitively, physically realizable drawings cannot contain all transitivity constraints in (4) because that would preclude valid drawings such as the one on the right side of Fig. 2. However, by definition, a physically realizable drawing needs transitivity enforced on disks that intersect to form a face, giving rise to (12).

Let $\mathcal{F}_{PR}$ be the formulation with constraints (2), (3), (12), (5), and (6). Given a solution satisfying $\mathcal{F}_{PR}$ we can build a solution satisfying (2) as equality, as stated in Proposition 1. If we think of w as relations between disks, solutions satisfying $\mathcal{F}_{PR}$ are partial orders between the disks of S_f for each $f \in F$. Proposition 1 shows that we can transform them into total orders without decreasing the objective function value.

Proposition 1 *Given a solution satisfying $\mathcal{F}_{PR}$, we can build a solution also satisfying (2) as equality with greater or equal objective value.*

Proof First, let us restrict ourselves to S_f for each $f \in F$. Define a digraph $K_{S_f} = (V, A)$, with vertex set corresponding to the disks in S_f, where $v(d)$ is the vertex correlated to disk d. There is an arc $(v(i), v(j))$ in A iff $w_{ij} = 1$. Clearly, this graph is acyclic. Let $h(v)$ be the position of vertex v in some topological order of K_{S_f}. Then, for each pair of disks $i, j \in S_f$ such that $w_{ij} = 0$ and $h(v(i)) > h(v(j))$, we set $w_{ij} = 1$ and add $(v(i), v(j))$ to A. One can see that K_{S_f} remains acyclic and thus w satisfies (12). We also now have that either $w_{ij} = 1$ or $w_{ji} = 1$ for all pair of disks $i, j \in S_f$. It is clear that for each pair $i, j \in S$, every w_{ij} or w_{ji} will be set to 1, except for those pairs such that both w_{ij} and w_{ji} were initially set to 0 and $\{i, j\} \notin S_f$ for any $f \in F$. But for these pairs, we may arbitrarily set any of them to 1, since this will not violate any transitivity constraint. Hence, this new solution satisfies (2) as equality. Because no w variable was set to 0 during this process, no x variable has decreased in value (see (3)). Therefore, the objective function value cannot go down. □

We thus define an alternative formulation $\mathcal{F}'_{PR}$ that is equivalent to $\mathcal{F}_{PR}$ with (2) replaced by an equality. We now show that we can solve the Max–Total physically realizable drawing problem by solving the ILP model consisting of maximizing (1) subject to $\mathcal{F}'_{PR}$, through Proposition 2.

Proposition 2 *The solution that maximizes (1), subject to $\mathcal{F}'_{PR}$ is an optimal solution for the Max–Total physically realizable drawing problem.*

Proof It suffices to show that a solution that satisfies $\mathcal{F}'_{PR}$ and maximizes (1) corresponds to a physically realizable drawing and that, conversely, any physically realizable drawing corresponds to a solution that satisfies $\mathcal{F}'_{PR}$.

Let (x^*, w^*) be a solution satisfying $\mathcal{F}_{PR}$ that maximizes (1). We first note that for each $f \in F$, there exists a total order between disks in S_f induced by w^*, since this relationship between disks in S_f is antisymmetric and total due to (2) and transitive due to (12). Also, there is no two disks i, j that have their relative order swapped for different faces, because in this case we would have $w^*_{ij} = w^*_{ji} = 1$, contradicting the fact that w^* is antisymmetric. This means that the orders induced by faces do not conflict and it is physically possible to draw them with these orderings. Also, because we are maximizing (1), any arc that would be visible in such drawing is set to 1. If an arc r is not visible in the drawing, it means there is a disk j that contains it and is above d_r, so $w^*_{d_r j} = 0$ and $x^*_r = 0$.

Given a physically realizable drawing, we consider the total order induced by each face f from the arrangement. Given two disks $i, j \in S_f$, we assume, w.l.g., that i is above j. We then set $w_{ij} = 1$ and $w_{ji} = 0$. It clearly satisfies (2) as equality. For any three disks $i, j, k \in S_f$, assuming i is above j and j is above k, it is true that i is above k, and thus such construction satisfies (12) for f. For pairs of disks i, j that do not both belong to any S_f, we arbitrarily set $w_{ij} = 1$

and $w_{ji} = 0$. Since, in this case, w_{ij} does not appear in (12), it is enough to observe that it satisfies (2) as equality. Given any visible arc r from this drawing, its disk d_r must be above all disks containing r, so setting $x_r = 1$ will satisfy (3). □

3.3 The Max–Min PRDP

In this section, we show how to modify our formulation for the Max–Total PRDP to solve the Max–Min version. We add a real variable z and the following constraints:

$$z \leq \sum_{r \in R | d_r = i} x_r \ell_r, \quad \forall i \in S \tag{13}$$

For a given disk $i \in S$, the right-hand side of (13) is b_i, as defined in Sect. 1. If z satisfies (13) and the objective function is set to maximize z, we obtain $z = \min\{b_i | i \in S\}$, which is the Max–Min metric.

3.4 Additional inequalities

In a physically realizable drawing, given two arcs r_1 and r_2, if d_{r_1} contains r_2 and d_{r_2} contains r_1, then at most one of these arcs can be visible. By using graph G_I as in Sect. 3.1, we see that (7) is also valid for our model.

From Fig. 7, note that for a nondegenerated vertex v there is a face contained in the same disks as v. This implies a total order among such disks and, therefore, locally around v, these disks behave as a stacking. Because the possible configurations around v are the same for stacking and physically realizable drawings, (8)–(11) are also valid for our model.

We now derive polyhedral properties for PRPD. Let us denote the convex hull of the points satisfying $\mathcal{F}_{PR}$ by P_{PR}, and establish some of its theoretical properties.

Define P_{SD} as the convex hull of feasible solutions of the (2)–(6). Since $\mathcal{F}_{PR}$ contains a subset of the constraints that define P_{SD}, the dimension of P_{PR} must be greater than or equal to the dimension of P_{SD}. Because the latter has full dimension [6], so does P_{PR}. Therefore, any facet-defining inequality for P_{SD} that is valid for P_{PR} is also facet-defining for P_{PR}. Thus, (2), (3), (12), and (7) define facets of P_{PR}, because (2)–(4) and (7) define facets of P_{SD} [6].

4 Decomposition techniques

In addition to the trivial decomposition that considers disjoint sets of disks independently, two decomposition techniques are presented in [8], which we briefly describe here. First, observe that if a disk d_1 is contained inside another disk d_2, there exists an optimal solution in which d_1 is drawn above d_2. In general, if two sets of disks do not intersect at their boundaries, such as sets $\{a, b\}$ and $\{c, d, e, g\}$ in Fig. 8(i), the drawing problem can be solved independently

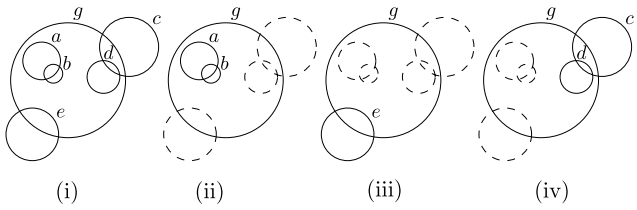


Fig. 8 An instance that allows for decomposition

for each set. To combine those solutions, sets of disks contained inside other disks (e.g., $\{a, b\}$ are inside g) can be drawn above the disks containing them, while keeping the orders resulting from the independent solutions. These arguments hold for both Max–Total and Max–Min problems.

Given a set of disks S, we can define a disk graph, $G_S = (V, E)$, with a vertex $v(d) \in V$ corresponding to a disk $d \in S$ and an edge $(v(d_1), v(d_2))$ belonging to E if disks d_1 and d_2 overlap. If this graph is not biconnected, there must exist an articulation point $v(d^*)$ in it. The removal of the corresponding disk d^* from S will spawn new connected components in G_S. It is a simple exercise to verify that if we replicate d^* in each set of disks corresponding to these components, then these augmented sets may be solved separately and their solutions easily assembled into an optimal solution to the original Max–Total problem. In our example, this corresponds to the instances in Figs. 8(ii)–(iv). On the other hand, this method may produce suboptimal solutions if used for the Max–Min objective. To see that, consider the example in Fig. 6(i) and its disk graph depicted in Fig. 10(i). Since d is an articulation point, this decomposition method splits the graph in several components that are solved to optimality as shown in Fig. 10(ii). By combining these optimal solutions, one ends up with the drawing shown in Fig. 6(ii), which clearly does not have the same Max–Min value as the optimal solution in Fig. 6(iii).

We now introduce a new decomposition technique that takes advantage of the specific structure of the Max–Total PRDP. If a pair of disks i and j are such that any face of the arrangement contained in both of them is not contained in any other disk, then all induced orders that include i and j are restricted to these two disks. Hence, any order we choose between i and j will not conflict with any other induced order. Therefore, we may remove the corresponding edge $(v(i), v(j))$ from G_S, solve for the connected components of G_S independently, and later decide the relative order between i and j, in a greedy way. For example, Fig. 9(i) depicts a set of disks with the underlying disk graph. The dashed edges can be removed from G_S. However, because face f is contained in d_1, d_2 and d_3, no edges between vertices corresponding to those disks may be removed. The resulting connected components are $\{d_1, d_2, d_3\}$, $\{d_4\}$ and $\{d_5\}$. This decomposition is not guaranteed to produce optimal solutions for the Max–Min objective. To see that, we

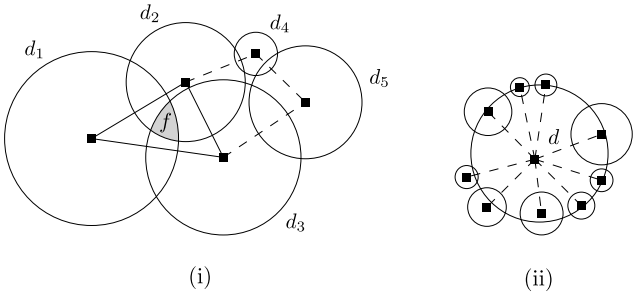


Fig. 9 (**i**) An instance with its corresponding disk graph. *Dashed edges* may be ignored in the ILP model. (**ii**) An instance where disk d would be replicated nine times when applying the decomposition that removes articulation points

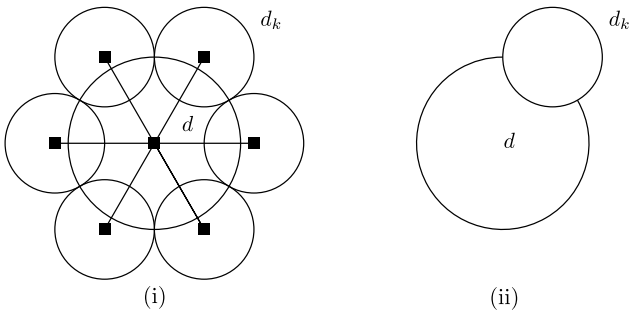


Fig. 10 (**i**) Disk graph of the instance in Fig. 6(**i**) and (**ii**) a Max–Min optimal solution for disks d and d_k

refer again to the instance in Fig. 6(i). According to this decomposition, we would remove all the edges of the disk graph shown in Fig. 10(i) and solve the problem by simply computing the optimum of the instances formed by the central disk and each of the surrounding ones. However, in these two-disk instances, the smaller disk goes on top of the larger one as illustrated in Fig. 10(ii), leading us to the drawing of Fig. 6(ii) which, as seen before, does not optimize the Max–Min value.

5 Implementation details

Our implementation was done in C++ (gcc 4.4.3) and used CGAL [2] v3.5.1 to gather the necessary input data. We used the MILP solver XPRESS [3] v20.00.05 to solve the optimization models and ran our experiments on an Intel Core 2 Quad 2.83 GHz machine with 8 GB of RAM, under GNU/Linux v2.6.32.

ILP/MILP models Both of our original optimization models include (2) as an equality, (3), (5), (6), and (12).

When it comes to (7), because the number of maximal cliques in a graph may be an exponential, we decided to select only some of them using the following heuristic. For each face f, let B_f^+ initially be the set of arcs r that belong to the boundary of f and whose disks d_r contain f. Let C_f^+ initially be the set of all disks that contain an arc in B_f^+. It is easy to see that the vertices corresponding to disks in C_f^+ form a clique in G_I. Since this clique is not necessarily maximal, we might try to extend C_f^+ (and its corresponding clique). Let r' be an arc contained in all disks in C_f^+ and whose disk $d_{r'}$ contains all arcs in B_f^+. The vertex set corresponding to $C_f^+ \cup \{d_{r'}\}$ forms a clique in G_I. We thus add $d_{r'}$ to C_f^+ and r' to B_f^+, and repeat this procedure until the resulting clique is maximal in G_I.

Surprisingly, our experiments showed that extending the original set C_f^+ decreases the performance of the branch-and-bound algorithm. One possible explanation is that this extension increases the density of the model (in terms of its coefficient matrix), making it harder to solve at each search node. Therefore, we opted for simply using the initial C_f^+ in the experiments reported in Sect. 7. As a consequence, the total number of constraints (7) and (8)–(11) is relatively small when compared to the number of constraints in the original model (16.3 % on average). Hence, we included all of those constraints at the beginning of the search, instead of using a separation procedure to add them gradually as they became violated (a practice known as *branch-and-cut*).

Max–Min PRDP heuristics As mentioned in Sect. 1.1, Cabello et al. developed an $O(n^2 \log n)$ algorithm for the Max–Min stacking drawing problem. Since a stacking drawing is a special case of a physically realizable drawing, we may use this algorithm to obtain a feasible solution for the Max–Min PRDP. Feasible solutions provide lower bounds for maximization problems that, when tight, may improve the performance of the branch-and-bound algorithm.

The polynomial-time algorithm can also be used for the physically realizable version when the decompositions of Sect. 4 create more than one component. Let n_C be the number of components and m_i be the optimal value of component $1 \leq i \leq n_C$ for the Max–Min stacking drawing problem. Let the labels of the components be such that $m_1 \leq m_2 \leq \cdots \leq m_{n_C}$, which means that the optimal solution for the stacking drawing version for the whole instance is m_1. Now, let m'_i be the optimal value of component i for the physically realizable version. It is easy to see that if $m'_i \leq m_{i+1}$, $1 \leq i < n_C$, then the optimal value for the Max–Min physically realizable drawing problem is $\min_{j \leq i}\{m'_j\}$. Therefore, we first presolve each component i using the Max–Min algorithm for stacking drawings to obtain m_i. We then solve for the physically realizable version of each component in nondecreasing order of m_i to obtain m'_i, and as soon as we find that $m'_i \leq m_{i+1}$, we can stop.

XPRESS parameters For reproducibility purposes, we provide the XPRESS parameters that had their default values changed in our experiments: `XPRS_MIPRELSTOP`

Table 1 Decomposition results

Instance	Disks	Decomp. ACB			Decomp. AB		
		Max	#	Avg.	Max	#	Avg.
City 156	156	26	66	2.39	29	53	3.09
City 538	538	53	258	2.10	53	240	2.35
Death	573	70	355	1.62	70	333	1.77
Magnitude	491	50	116	9.92	45	116	11.22

set to 0.0, `XPRS_MIPABSSTOP` set to 10^{-7}, `XPRS_MIPRELCUTOFF` set to 0.0, `XPRS_MIPADDCUTOFF` set to 10^{-7}, and `XPRS_MAXTIME` set to -18000. For more information on these parameters and their default values, please refer to the XPRESS-Optimizer Manual [3].

6 Problem instances

We assess the effectiveness of our solution approach through a series of experiments with various data sets. From [1], we use the following datasets: *City 156* and *City 538* (populations of the 156 and 538 largest cities in the United States); and *Earthquake-Death* and *Earthquake-Magnitude* (death counts and magnitudes associated with earthquakes around the world).

In addition to the data sets above, we created additional instances consisting of the populations of the largest cities in the following countries: Australia, Belgium, Brazil, China, Denmark, France, Germany, Indonesia, Israel, Italy, Japan, Netherlands, Norway, Portugal, Spain, Russia, UK, and eastern/western USA.

The instances used in our experiments and all the known optimal solutions are available for download [5].

7 Results and discussion

Decomposition results We begin by discussing the effects of the decomposition techniques on the instances used in [1], which are summarized in Table 1. For simplicity, we name the decompositions as follows. Decomposition A is that which regards sets of disks with no boundary intersection independently; decomposition B is the one that keeps removing articulation points until the resulting components are biconnected; and decomposition C is the new one introduced in Sect. 4. We reproduced the results in [8] by decomposing the original instance with A, and further decomposing each resulting component with B. We denote this chain decomposition as AB. Using similar notation, we denote by ACB the decomposition sequence of A, followed by C, and then B. The reason to perform C before B is that some cases are decomposable by either B or C, as in Fig. 9(ii), but since B replicates vertices, increasing the total number of disks to be solved, it is best to apply C first.

The first two columns of Table 1 indicate the names of the instances and their original number of disks. For each decomposition, we show the size of the largest component (Max), the total number of resulting components (#), and the average component size (Avg.) obtained after performing the decomposition. Multiplying the average number (of disks) by the number of instances will not necessarily produce the number of original disks because decomposition B replicates disks.

The reductions in problem size are remarkable. For example, instances City 538 and Magnitude can now be solved by optimizing over sets of disks no larger than about a tenth of their original sizes. Even after introducing decomposition C, the largest component of most instances remained unbroken, but this decomposition did split other smaller components, decreasing the average number of disks to be solved at a time.

Table 2 Results on the largest components that were solved from each original problem instance. Times are in seconds

Component	Base value	Optimal value	Nodes		Time	
			PR	SD	PR	SD
538-1-6 (29)	21.98	44.32	1	1	3	5
538-1-0 (51)	77.37	90.08	1	1	4	19
538-24-0 (53)	18.98	65.08	177	453	14554	84308
death-2-0 (70)	725.28	1152.13	1	1	6	61
mag-6-0 (26)	217.21	579.58	1	1	4	13
mag-1-1 (39)	417.32	1128.52	1	1	13	48
mag-5-0 (81)	601.79	1914.27	1	1	14	2312
mag-1-0 (113)	581.41	3158.82	3	1	107	34306
mag-7-0 (116)	700.37	2916.17	1	1	42	25256

Experimental results with the Max–Total PRDP We focus on the challenging components from City 538, Earthquake-Death and Earthquake-Magnitude because the remaining instances/components could be solved very easily. In general, instances whose G_S graphs contain large cliques tend to be the most challenging for our algorithm. Table 2 summarizes our results.

The first column contains the component name in the form α-β-γ (δ), where α relates to the original instance ("538" for City 538, "death" for Earthquake-Death and "mag" for Earthquake-Magnitude), β identifies the component id from decomposition A, and γ indicates the γ-th component obtained after performing decomposition B on component β. Finally, δ denotes the number of remaining disks in this component. Column *Base Value* shows the total length of the arcs that are always visible in any solution,

Table 3 Optimal solutions for population instances

Country	Base value	Optimal solution	
		(PR)	(SD)
Belgium (312)	5354.300	3127.988	3127.788
China (141)	1988.467	2409.173	2409.151
Denmark (310)	4640.934	2301.315	2301.089
Indonesia (150)	2062.608	1275.765	1275.749
Israel (150)	1772.046	1892.881	1892.823
Netherlands (367)	6459.026	4720.454	4720.453
Norway (150)	2108.152	1230.658	1230.632
Spain (300)	4469.564	3861.171	3861.157
UK (186)	2530.148	1999.492	1999.491
US (East) (150)	1810.834	2462.580	2462.556

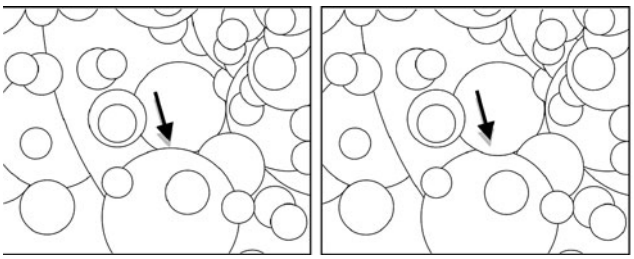


Fig. 11 A close up of the Denmark instance, showing a difference between the SD and PR optimal solutions

that is, those that are not contained in any disk. The *Optimal Value* column shows the value of the optimal solution minus the base value. The last four columns show the number of search nodes and time (in seconds) required by our algorithm (PR) and by the algorithm in [6] (SD). In all instances, PR requires less time to obtain provably optimal solutions than SD does, sometimes by more than one order of magnitude.

Tables 3 and 4 show results for ten population-based instances for which an optimal physically realizable drawing has an objective value strictly greater than the value of an optimal stacking drawing, thus resulting in visually better solutions. See Fig. 11 for an example.

Table 3 shows the base values as well as the optimal solution values obtained by the physically realizable (PR) and the stacking drawing (SD) algorithms, respectively. Table 4 complements Table 3 with the number of nodes explored by each algorithm and their execution times. Table 5 extends Table 4 for instances whose optimal PR and SD solutions have the same value.

Although the solution values of the physically realizable drawings in Table 3 are only slightly greater than their stacking counterparts, when looking at a map with hundreds of disks, even minor improvements can mean the difference between seeing or missing a city.

Table 4 Search nodes and time for population instances

Country	PR		SD	
	Nodes	Times	Nodes	Times
Belgium (312)	18	6	18	2757
China (141)	5	6	5	3797
Denmark (310)	19	9	19	1007
Indonesia (150)	2	14	2	2477
Israel (150)	6	22	6	1044
Netherlands (367)	23	9	24	4222
Norway (150)	4	11	3	3433
Spain (300)	16	54	15	194
United Kingdom (186)	8	7	8	422
United States (East) (150)	8	22	6	1084

Table 5 Search nodes and time for other countries

Country	PR		SD	
	Nodes	Times	Nodes	Times
Australia (210)	10	4	10	3250
Brazil (150)	10	45	10	787
Canada (150)	10	17	8	950
France (135)	8	1785	8	2288
Germany (150)	16	74	12	737
Italy (300)	22	135	23	538
Japan (150)	6	153	6	2005
Portugal (150)	4	486	4	2525
Russia (150)	3	8	3	690
United States (West) (87)	6	704	6	2788

In terms of execution times, once again, the PR algorithm is greatly superior to the SD algorithm, as shown in Tables 4 and 5. As before, improvements can range from one to more than two orders of magnitude.

Experimental results for the Max–Min PRDP We now present the results for the Max–Min PRDP using the challenging components from City 538, Earthquake-Death, and Earthquake-Magnitude as input. Table 6 shows our results for the components that had the same optimal Max–Min values for both stacking and physically realizable drawings.

In general, finding the optimal Max–Min drawing is computationally more challenging than finding the optimal Max–Total drawing in the physically realizable case. Component 538-24-0 was the most challenging for our Max–Min PR algorithm, which was not able to find an optimal solution even after 10 CPU hours. The value reported for this component in Table 6 was obtained using the Max–Min SD algorithm. Table 7 has results for the remaining components, which had better Max–Min values for physically realizable drawings.

Table 6 Results on the largest components that were solved from each original problem instance. Times are in secs

Component	Value	Times
538-1-6 (29)	1.22	3
538-1-0 (51)	1.15	4
538-24-0 (53)	0.76	>10 h
death-2-0 (70)	0.77	8
mag-6-0 (26)	12.64	15
mag-1-1 (39)	14.30	82
mag-1-0 (113)	12.87	1536

Table 7 Results on the largest components that were solved from each original problem instance. Times are in secs

Component	SD value	PR	
		Value	Times
mag-5-0 (81)	13.59	13.61	1919
mag-7-0 (116)	12.01	13.05	4802

For the population-based instances, we also have optimal solutions with the same Max–Min value for both physically realizable and stacking drawings. Column *Value* in Table 8 shows the optimal values. For these instances, we used the heuristic mentioned in Sect. 5 in order to solve fewer components. We were able to find all of the optimal solutions solving only the first analyzed component of each instance. The last three columns of Table 8 show, respectively, the number of components, the size of the solved component, and the execution time of the algorithm in seconds.

Even using heuristics to boost computation, some instances are still hard to solve, such as France, which took more than an hour and USA West, which was not solved to optimally after 10 hours of execution.

Experiments with Max–Total stacking drawings Since the PR algorithm is much faster than its SD counterpart [8], we decided to adapt it to the Max–Total stacking drawing problem (SDP) as follows. We first run the PR algorithm on a given instance. If the optimal solution has no cyclic order among the disks, the physically realizable drawing is also a stacking drawing, and we are done. Otherwise, we choose any cycle in the solution and add a transitivity constraint for each triple of vertices in this cycle. We then resolve the instance with these additional constraints, repeating until no cyclic order exists. In this procedure, decomposition C, as defined in the beginning of this section, cannot be used.

Table 9 shows the results of this new method on the population-based instances of Table 3. The second and third columns show the number of cycles n_C that were removed by the algorithm, and the size of the largest removed cycle

Table 8 Optimal solutions for population instances: Max–Min PR

Country	Value	Components		Times
		#	Size	
Australia (210)	6.41	62	3	2
Belgium (312)	13.92	69	145	5
Brazil (150)	13.44	35	88	68
Canada (150)	9.50	29	87	17
China (141)	10.34	8	123	9
Denmark (310)	8.79	70	102	4
France (135)	6.45	53	41	4826
Germany (150)	9.60	37	61	182
Indonesia (150)	7.58	33	99	43
Israel (150)	11.49	20	54	6
Italy (300)	6.18	39	67	500
Japan (150)	8.73	29	101	682
Netherlands (367)	15.72	63	225	9
Norway (150)	6.73	35	98	15
Portugal (150)	9.14	30	46	59
Russia (150)	7.42	23	114	12
Spain (300)	8.73	41	46	99
United Kingdom (186)	9.03	25	135	9
United States (East) (150)	9.34	8	85	144
United States (West) (87)	9.31	6	76	>10 h

Table 9 Optimal solutions for population instances: SD

Country	Cycles removed		Times	Ratios w/SD
	n_C	$C_{\max}$		
Belgium (312)	2	4	6	459.5
China (141)	1	8	6	632.8
Denmark (310)	4	4	15	67.1
Indonesia (150)	1	4	17	145.7
Israel (150)	2	4	23	45.4
Netherlands (367)	1	4	8	527.8
Norway (150)	1	3	15	228.9
Spain (300)	2	9	49	4.0
UK (186)	1	5	7	60.3
US (East) (150)	1	4	25	111.5

$C_{\max}$. Because the algorithm adds a new constraint for each triple of vertices in a cycle, at most $n_C \cdot C_{\max}^3$ constraints are added to the original model. The execution times (in seconds) appear in the fourth column.

As expected, the number of cycles and their length were very small when compared to the total number of disks. Even though we had to solve the model $n_C + 1$ times, our method is still faster than running the original SD algorithm, as seen in the last column of Table 9, which has the ratio be-

Table 10 Results on the instances that have the same optimal value for both types of drawings, but for which the PR algorithm returns a non-stacking drawing. Times in secs

Country	Cycles removed		Times	Ratios w/SD
	#	Max size		
538-1-6 (29)	1	5	3	1.7
Magnitude-7-0 (116)	1	3	65	388.6
France (135)	1	3	1783	1.3

tween the execution time of the SD algorithm and that of our method.

When the SD and PR algorithms return optimal solutions with the same value, it is not necessarily true that the solution returned by the PR algorithm is a stacking drawing. This can be seen in Table 10 which shows results for instances in Tables 2 and 5 that had cyclic orders when solved by the PR algorithm.

For the remaining instances, the PR and SD execution times were not considerably different.

8 Conclusion

Proportional symbol maps (PSMs) are an important visualization tool for geopositioned data. The symbols we consider are opaque disks whose areas are proportional to the magnitude of the data they represent, and we focus on the class of physically realizable drawings.

Visually effective PSMs typically maximize one of two quality metrics: the total length of the visible borders of the disks on the map (Max–Total), or the minimum length of the visible disk borders (Max–Min); both of which are known to be NP-Hard. Physically realizable drawings can be superior to the previously studied stacking drawings because they have the potential to expose greater portions of the disk borders.

We propose and implement exact algorithms to solve the Max–Total and Max–Min problems. Our optimization approach is based on integer programming formulations enhanced by the application of known and novel decomposition techniques, as well as the introduction of several families of strong inequalities.

Our computational results, which involve real life data sets related to natural events and population statistics, indicate that, in addition to being visually superior, optimal physically realizable drawings can be obtained at a fraction of the computational effort required to obtain optimal stacking drawings when optimizing the Max–Total metric.

To the best of our knowledge, we are the first to find provably optimal physically realizable drawings for the data sets proposed in [1], as well as the population-based data sets described in Sect. 6.

Finally, we also implement a method to solve the Max–Total stacking drawing problem using the above algorithm and additional constraints, which turns out to be faster than the algorithm proposed in [8].

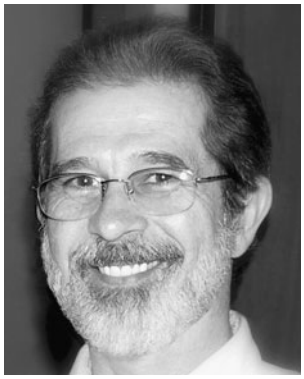
Acknowledgements Guilherme Kunigami is supported by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) grant 830510/1999-0. Pedro J. de Rezende is partially supported by CNPq grants 483177/2009-1, 473867/2010-9, FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) grant 07/52015-0, and a grant from FAEPEX/UNICAMP. Cid C. de Souza is partially supported by CNPq grants 301732/2007-8, 472504/2007-0, 473867/2010-9, and FAPESP grant 07/52015-0.

The authors would like to express their appreciation to the anonymous referees for their careful and thorough reviews and for their comments which contributed to improving the exposition.

References

1. Cabello, S., Haverkort, H., van Kreveld, M., Speckmann, B.: Algorithmic aspects of proportional symbol maps. Algorithmica **58**(3), 543–565 (2010)
2. CGAL: Computational Geometry Algorithms Library. www.cgal.org
3. Fair Isaac Corp: Xpress optimizer reference manual
4. Griffin, T.: The importance of visual contrast for graduated circles. Cartography **19**(1), 21–30 (1990)
5. Kunigami, G., Cano, R.G., de Rezende, P.J., Yunes, T.H., de Souza, C.C.: Proportional symbol maps—benchmark instances (2011). www.ic.unicamp.br/~Cid/Problem-instances/Symbol-Maps
6. Kunigami, G., de Rezende, P.J., de Souza, C.C., Yunes, T.: Optimizing the layout of proportional symbol maps. www.optimization-online.org/DB_HTML/2010/11/2805.html
7. Kunigami, G., de Rezende, P.J., de Souza, C.C., Yunes, T.: Determining an optimal visualization of physically realizable symbol maps. In: Lewiner, T., Torres, R. (eds.) Proc. of the 24th Conf. on Graphics, Patterns and Images. IEEE Comp. Soc. Conf. Pub. Serv., pp. 1–8 (2011)
8. Kunigami, G., de Rezende, P.J., de Souza, C.C., Yunes, T.: Optimizing the layout of proportional symbol maps. In: Murgante, B. et al. (eds.) Proceedings of ICCSA 2011. Lecture Notes in Computer Science, vol. 6784, pp. 1–16. Springer, Berlin (2011)
9. Slocum, T.A., McMaster, R.B., Kessler, F.C., Howard, H.H.: Thematic Cartography and Geographic Visualization, 2nd edn. Prentice Hall, New York (2003)
10. Wolsey, L.A.: Integer Programming. Wiley, New York (1998)

Guilherme Kunigami is a software engineer working with Operations Research. His research interests are in combinatorial optimization (including integer linear programming and metaheuristics), artificial intelligence, computational geometry, and computer graphics. He holds a M.Sc. degree in Computer Science and B.S. degree in Computer Engineering from the University of Campinas, Brazil.



Pedro J. de Rezende received B.Sc. and M.Sc. degrees in Mathematics from the University of Brasília, Brazil, in 1977 and 1979. In 1988, he received his Ph.D. in Computer Science from Northwestern University, USA. He was a faculty member at the University of Brasília, Brazil, 1978–1979; Northeastern University, USA, 1985–1988; Wellesley College, USA, 1989–1990; and has worked at University of Campinas, Brazil, since 1990. In 2006–2007, he visited McGill University, Canada. His research interests include computational geometry, combinatorial optimization, pattern recognition, and medical imaging applications.

Cid C. de Souza got his Ph.D. degree in Applied Sciences at the Center for Operations Research and Econometrics of the Catholic University of Louvain, Belgium, in 1993. He also holds M.S. and B.S. degrees in electrical engineering from the Catholic University of Rio de Janeiro, in Brazil. He is currently a full professor at the Institute of Computing of the University of Campinas, Brazil. His main research interests include combinatorial optimization, linear and integer programming, and design and analysis of algorithms.

Tallys Yunes is an Associate Professor of Management Science at the University of Miami School of Business Administration. His research interests are in applied optimization, including scheduling, product line simplification, transportation, visualization, and location problems. He holds a Ph.D. in Operations Research from Carnegie Mellon University, as well as M.S. and B.S. degrees in Computer Science and Computer Engineering from the University of Campinas, in Brazil.