Algorithms for Labeling Focus Regions

Martin Fink, Jan-Henrik Haunert, André Schulz, Joachim Spoerhase, and Alexander Wolff

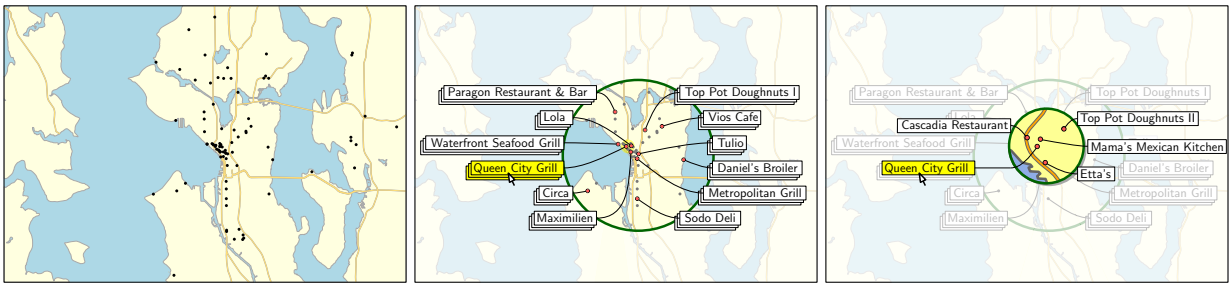


Fig. 1. In a map of Seattle (left) showing 86 restaurants (black dots), a user selects a focus region (middle). At the boundary of that region (green circle), labels are placed for a selection of the restaurants; curved lines connect the labels with the actual sites of the restaurants. Every labeled restaurant represents a cluster of restaurants, which we indicate by drawing a stack of rectangles with the label on top; when clicking a label, a detailed labeling for the corresponding cluster pops up (right).

Abstract—In this paper, we investigate the problem of labeling point sites in focus regions of maps or diagrams. This problem occurs, for example, when the user of a mapping service wants to see the names of restaurants or other POIs in a crowded downtown area but keep the overview over a larger area. Our approach is to place the labels at the boundary of the focus region and connect each site with its label by a linear connection, which is called a leader. In this way, we move labels from the focus region to the less valuable context region surrounding it. In order to make the leader layout well readable, we present algorithms that rule out crossings between leaders and optimize other characteristics such as total leader length and distance between labels. This yields a new variant of the boundary labeling problem, which has been studied in the literature. Other than in traditional boundary labeling, where leaders are usually schematized polylines, we focus on leaders that are either straight-line segments or Bézier curves. Further, we present algorithms that, given the sites, find a position of the focus region that optimizes the above characteristics. We also consider a variant of the problem where we have more sites than space for labels. In this situation, we assume that the sites are prioritized by the user. Alternatively, we take a new facility-location perspective which yields a clustering of the sites. We label one representative of each cluster. If the user wishes, we apply our approach to the sites within a cluster, giving details on demand.

Index Terms—Focus+context techniques, data clustering, mobile and ubiquitous visualization, geographic/geospatial visualization.

1 Introduction

Users of maps normally expect answers to specific queries, for example, where to find a good restaurant or how to reach a certain destination. General-purpose topographic maps do not answer such queries satisfactorily, thus have become almost obsolete. Instead, Internet mapping services such as Google Maps or Bing Maps offer interfaces that allow for user interactions and sophisticated map visualization. Still, the existing systems do not fully support focus-and-context visualization, which generally aims at emphasizing regions and themes of interest while showing overview information for orientation.

In order to emphasize a focus region (for example, the user's vicinity), many researchers have proposed to (locally) increase the map scale in that region, which allows more details to be presented. Different methods have been proposed to define a seamless transition between a large-scale focus region and a small-scale context region, including fish-eye projections [1] and, for network maps, optimization-based graph drawing methods [2]. Another common approach is to use *portals* [3], that is, windows with detailed information superimposed on an overview map. Additionally, emphasis can be put on map objects by appropriately setting their colors [4].

Considering that the focus and the context region of a map serve different purposes, we argue that special labeling techniques are needed for focus-and-context maps. Map space can be regarded as a resource that is more expensive in the focus region, thus text annotations and iconic labels for points of interest (POIs) or *sites* in the focus region should preferably be moved to the context region. The difficulty herein is that correspondences between labels and sites have to remain clear. One possibility to achieve this is to display such correspondences as linear connections called *visual links* [5] or *leaders* [6]. We use the latter term, which is more common in the literature on map labeling.

In this paper, we apply *boundary labeling* to focus-and-context maps. Boundary labeling commonly means to place labels at the boundary of a *map* and, for each label, to draw a leader that connects a point—called *port*—on the label boundary with the corresponding site in the map. In a focus-and-context map, we suggest to place labels at the boundary of the *focus region*, which may be given explicitly as part of a user's query. This is the case in the scenario shown in Fig. 1, where a user specifies a circular region in order to query the restaurants in that region. If the user does *not* specify a focus region, however, a visualization system should still be able to produce a good focus-and-context map. As a general rule, focus should be put on regions with many interesting sites. Taking this rule into account, we develop also algorithms that determine a circular focus region of given radius such that the maximum number of sites is labeled given our boundary-labeling model.

The general problem with boundary labeling is that the leaders produce additional clutter and that the correspondence between labels and

- *M. Fink, J.-H. Haunert, J. Spoerhase, and A. Wolff are with Lehrstuhl I, Institut für Informatik, Universität Würzburg. WWW: http://www1.informatik.uni-wuerzburg.de/en/staff/*
- *A. Schulz is with Institut für Mathematische Logik und Grundlagenforschung, Universität Münster.*

Manuscript received 31 March 2012; accepted 1 August 2012; posted online 14 October 2012; mailed on 5 October 2012.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

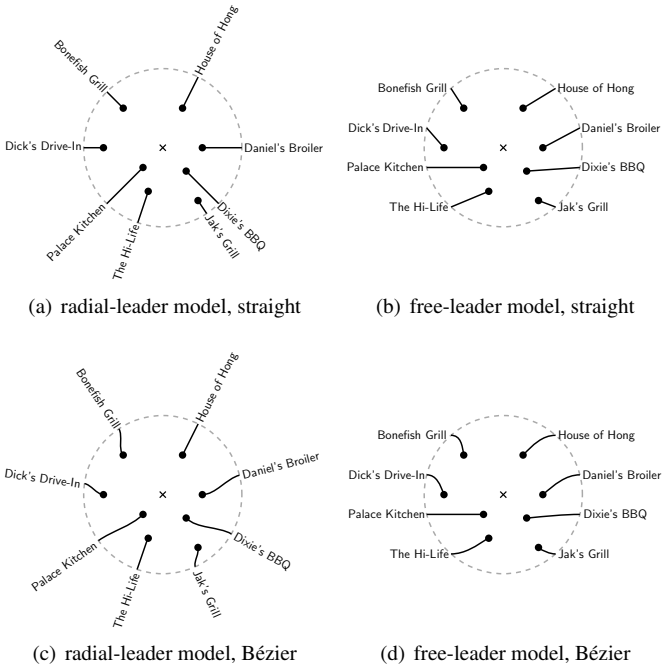
sites may become unclear, especially if leaders are zig-zagging, crossing each other, long, or close to each other. Therefore, we designed our boundary-labeling algorithms to avoid such unfavorable leader properties. For example, we consider a variant of the labeling problem where the leaders are straight-line segments, crossings are forbidden, and the total length of all leaders is minimized. We also present algorithms that visually improve solutions to this problem variant by transforming the straight-line segments into Bézier curves that have at most one inflection point. This allows us to control the slope of a leader in its site and port, for example, to ensure that a leader connects horizontally or vertically to its port and thus has the same slope as the boundary of its adjoining label. According to the Gestalt criterion of good continuation [7] this is favorable, as it allows map users to understand the label-site correspondences more easily.

Maps often contain too many points of interest to label them all. Therefore, in addition to choosing label positions and drawing leaders, we have to select a subset of the sites that will become labeled. An approach that is common in the literature on map labeling is to search for a maximum-cardinality set of sites that allows for a feasible labeling. If we have space for k labels at the boundary of the focus region, however, we can select *any* subset of k sites to become labeled. Among these solutions we may search for a labeling with short leaders, but this means that mainly sites at the boundary of the focus region will become selected. This is unfavorable, since normally users are particularly interested in the center of the focus region. Moreover, we argue that the selected subset should reflect the spatial distribution of all sites. If the input set contains a dense cluster (for example, a city center with many restaurants), the selected subset should also contain a cluster (which may be less dense) in the same area. In order to take this additional criterion into account, we suggest a novel approach by modeling map labeling as a facility location problem.

Our contribution is as follows.

- We discuss two leader models; the *radial-leader model* (see Fig. 2(a)) and the *free-leader model* (see Fig. 2(b)) [1] and formally state the corresponding layout problems (Sect. 3). While the problems concerning the free-leader model can be reduced to specific matching problems on graphs for which efficient algorithms are known, we present new algorithmic solutions to our problems in the radial-leader model (Sect. 4). Other than Plaisant and Fekete [8] who have used radial leaders before (see Fig. 3 and the discussion of previous work in Sect. 2), our leaders do not bend since we place the labels directly at the boundary of the focus region. In order to strengthen the visual connection between labels and objects, we deliberately decided not to place our labels as blocks of left-aligned text, but rather in the immediate vicinity of the focus region. While the approach of Plaisant and Fekete is meant for interactive exploration, ours also makes sense in a static environment where the aesthetic quality of the leader and label placement is crucial. For example, we optimize the layout in case there are more sites than space for labels. We are particularly interested in the algorithmic challenges behind these optimization problems.
- Among the algorithms for the radial-leader model, we also address a new optimization goal for circular focus regions: given the region's radius, we find a position of the region that maximizes the number of sites whose labels can be placed—without overlap—at the region's boundary; see Sect. 4.3
- We present two extensions (Sect. 5) that can be applied to both of our models, namely (i) a facility location model that allows us to simultaneously cluster and label a set of sites and (ii) a postprocessing for (non-crossing) straight-line leaders that transforms them into (non-crossing) Bézier curves; see Fig. 2 (bottom). To the best of our knowledge, neither clustering nor Bézier curves have been used for (boundary) labeling.
- We use the clustering from extension (i) to partition the focus region into subregions, one for each cluster. If the user clicks on the label that corresponds to the cluster (or into the subregion), we display the subregion enlarged and apply our labeling method to the sites in the subregion; see Fig. 1 (right) and Sect. 5.1.1.

[1] A video demo showing both models is available under `http://www1.informatik.uni-wuerzburg.de/pub/videos/infovis2012.mp4` and `http://vimeo.com/user12598215/circlelab`.



(a) radial-leader model, straight (b) free-leader model, straight

(c) radial-leader model, Bézier (d) free-leader model, Bézier

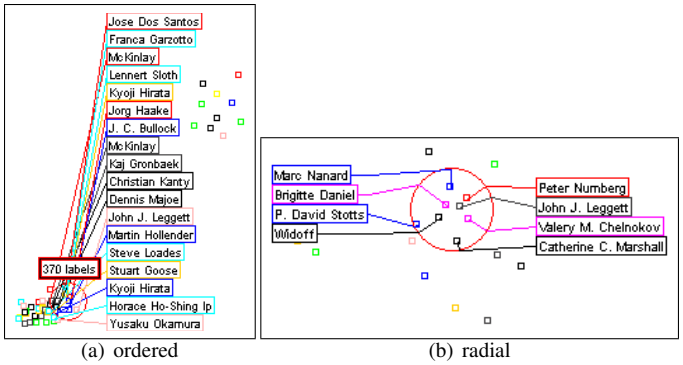
Fig. 2. Example labelings with our two leader models, top: drawn as straight-line segments, bottom: with Bézier postprocessing applied to the above straight-line solutions.

We present results of some preliminary experiments that we performed with implementations of our algorithms; the experiments and results are described in various sections of the paper.

2 Previous Work

Labeling geographic maps is a central problem in cartography. Labeling maps manually is a tedious task that, in the 1980's, was estimated to consume 50% of a map's production time [9]. Traditionally, labels are placed directly on the map; leaders are used very rarely. A simple model to establish a relationship between internal labels and sites is the four-position model [10, 11]. Here a label is represented by a rectangular box and the label is placed such that one of the four rectangle corners coincides with the associated site.

The idea to label data along a circular boundary has been applied before. The *excentric labeling* approach by Fekete and Plaisant [8] extends the infotip paradigm to label dense maps interactively. They draw a circular focus region of fixed radius around the current cursor position, and label the sites that fall into the circle. Labels are left-aligned in one or two stacks to the left and/or right of the circle, depending on where space is available. The labels are connected to the sites by leaders. For drawing the leaders, they present two main approaches. In the first approach, they insist on ordering the labels within each stack according to the vertical order of the corresponding sites. As a result, leaders may cross; see Fig. 3(a). In the second approach, a leader goes from a site via its projection on the focus circle and then, in the case of a right stack, to the mid-point of a left label edge. In the case of a left stack, a third segment may be needed in order to reach the right label edge of a very short label without introducing crossings between the leader of that label and other labels [8]. The authors call this approach the *non-crossing* or *radial* approach. Obviously, the higher the label stacks are the smaller the angles between the leaders get. If more sites lie in the focus circle than can be labeled, an arbi-



(a) ordered (b) radial

Fig. 3. Examples of excentric labeling. Clipped figures from [12].

trary subset of representatives is chosen and labeled; additionally, the number of sites in the circle is displayed (see Fig. 3(a)).

Fekete and Plaisant recommend the first approach as a default; the focus of their work was on interactive speed and on a comparison to statically labeled maps where users had to zoom and pan in order to find specific sites. They did a user study (with eight subjects) that showed that users completed the task using interactive labeling nearly twice as fast as when using static labeling. Fekete and Plaisant did not specify the asymptotic running times of their approaches, but it is easy to see that they run in $O(n \log n)$ time, where n is the number of points inside the focus circle.

Bekos et al. [6] introduced *boundary labeling* for labeling static maps. In this model, the "map" is contained inside a rectangle and labels are placed outside the rectangle. The model supports three types of leaders: straight-line segments and rectilinear polylines with one or two bends per line. Labels are either placed along one, two or all four sides of the boundary rectangle. The authors show how to construct a non-crossing labeling with minimal total leader length or minimum number of bends for some variants of their model. Later, many other variants of boundary labeling have been investigated, among others with octilinear leaders [13]. Recently, techniques for combining boundary labels with interior labels have been proposed [14].

Hartmann et al. [15] introduced a more general model for boundary labeling. In particular, they suggested a classification scheme that takes also more complex boundary shapes of the map into account, that is, the boundary can be a circle or more generally an arbitrary silhouette. A first algorithm for the silhouette scenario (based on a spring embedder) is due to Ali et al. [16].

Recently, Speckmann and Verbeek [17] introduced *necklace maps* to visualize statistical data on geographic domains. The size of the label is used to encode the value of the statistical variable and the labels are placed on circles or silhouettes. Necklace maps do not use leaders, but establish relations between geographic objects and labels by matching colors and proximity. This idea is effective if the geographic objects to be labeled have some spacial extent; here, in contrast, we assume that we are given point data.

3 Problem Statements and Motivations

We study several incarnations of the problem how to label focus regions. We distinguish two general models. In the first model, the focus region is a disk and each leader is a section of a ray that emanates from the center of the disk and connects a site to the boundary of the focus region; see Fig. 2(a). We refer to this model as the *radial-leader model*; we discuss it in Sect. 3.1. Note that if no two sites lie on the same ray, leaders are disjoint by construction. In the second model, the *free-leader model* (see Fig. 2(b) and Sect. 3.2), every port is placed on a prescribed position on the boundary of the focus region. We do not insist on any specific direction of the leaders; instead, we explicitly require leaders to not cross each other. Note that, in the free-leader model, there is no need to restrict oneself to circular focus regions. In fact, all results in this model hold for convex focus regions.

The choice of the model does not necessarily determine the orientation of the text labels. We think, however, that the radial-leader model suits radially oriented labels particularly well. In contrast, we suggest to use the free-leader model in conjunction with horizontally oriented labels. For examples, see Fig. 2 (top).

3.1 Radial Leaders

For the radial-leader model, we assume that the focus region is a disk $D = (c, r)$ with center c and radius r. Given a label ℓ, we define the port p_ℓ of ℓ by radially projecting the site s_ℓ labeled by ℓ onto the boundary ∂D of D, that is, p_ℓ is the intersection of ∂D with the ray that emanates from c and goes through s_ℓ.

Clearly, this model makes it difficult to accommodate labels if the projections of several sites lie in a very small part of ∂D. We model this by saying that two sites s and s' are *in conflict* if the angle $\angle scs'$ is smaller than a predefined value $\alpha > 0$. Our aim is to find (and label) a maximum-cardinality subset of the sites that is *conflict-free*, in the sense that no two sites in the subset are in conflict.

This model makes sense, for example, if (as in Fig. 2(a)) each label contains one line of text and has the same orientation as its leader. Correspondences between labels and sites are particularly easy to comprehend in this case, since the leaders can be perceived as continuations of their labels. On the downside, the user has to read rotated text. According to a user study [18], however, rotating a text by not more than 90 degrees leads to only small increases in the reading speed of users and the number of reading errors if the text consists of a single word with five to six letters. Therefore, we suggest to use this model if the label texts are not much longer than this.

Problem 1. *Label maximization with given center*
Input: *Disk $D = (c, r)$, set $S \subset D$ of n point sites, angle α.*
Output: *Maximum-cardinality conflict-free subset $S' \subseteq S$, that is, the angle formed by any two rays that emanate from c and go through points in S' is at least α and $|S'|$ is maximized.*

The radius r of D and the fixed font size determine the angle α.

Next we take priorities among the sites into account, which occur, for example, if some sites match a user query better than others. In the literature on map labeling, this is commonly modeled by defining weights for the sites and by selecting a maximum-weight set of sites that allows a feasible labeling.

Problem 2. *Weighted label maximization with given center*
Input: *Disk $D = (c, r)$, set $S \subset D$ of n point sites, weight function $w\colon S \to \mathbb{R}^+$, angle α.*
Output: *Conflict-free subset $S' \subseteq S$ such that the total weight $w(S') = \sum_{s \in S'} w(s)$ is maximized.*

For the remaining two variants of the radial-leader model, we support the user by finding a good position of the focus region. In many applications the focus is specified by a fixed set of sites rather than by a fixed focus region. In order to make sure that as much useful information as possible is displayed, we place the focus region such that it admits a maximum-cardinality conflict-free subset of sites.

Problem 3. *Label maximization with variable center position*
Input: *Set $S \subset \mathbb{R}^2$ of n point sites, angle α.*
Output: *Center $c \in \mathbb{R}^2 \setminus S$, maximum-cardinality subset $S' \subseteq S$ that is conflict-free w.r.t. c and α.*

For this problem, we also consider the weighted version, which we define analogously to Problem 2.

As the last problem of this section we ask for the disk center c, which maximizes the minimum angular separation of the sites w.r.t. c.

Problem 4. *Sector maximization*
Input: *Set $S \subset \mathbb{R}^2$ of n point sites.*
Output: *Center $c \in \mathbb{R}^2 \setminus S$ such that S is conflict-free w.r.t. the largest possible angle α.*

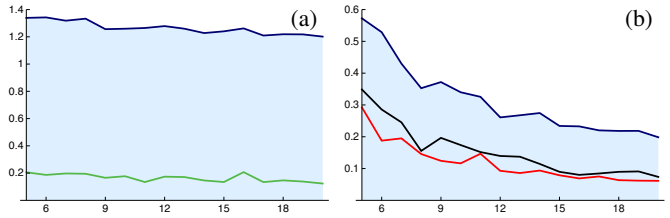


Fig. 4. (a) Radius of the smallest-angle focus disk relative to the smallest enclosing disk as measured in Experiment 1. The blue line shows the average ratio; the green line depicts the standard deviation in terms of the point set size n. (b) Average of the minimum angle relative to $360°/n$ of the smallest-angle focus disk (blue), center of gravity disk (black), and minimum enclosing disk (red) in terms of n as recorded in Experiment 2.

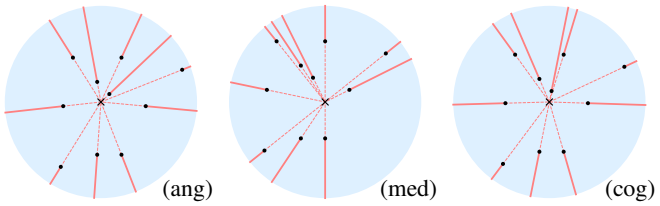


Fig. 5. An example point set with disk that maximizes the minimum angle between leaders (ang). For comparison: the minimum enclosing disk (med) and the disk centered at the center of gravity (cog). All three disks are drawn with the same radius.

Our main motivation to study Problem 4 stems from the case where *all* sites need to be labeled. Note that such a labeling exists if we make the radius r of the focus disk arbitrarily large enough. Therefore, we can assume that a small separating angle α suffices to have disjoint labels. In practice, however, we cannot arbitrarily increase r since the available space is limited. Therefore, it is reasonable to ask for the smallest disk that allows us to label all sites. If α is the smallest angle formed by two rays emanating from c and going through two sites in S, then r has to be greater than $1/\alpha$—assuming that labels have height 1. Therefore, minimizing the radius of the disk is equivalent to maximizing the smallest angle between any two rays.

Problem 4 also occurs when we have a solution to Problem 3, but two labels may be unnecessarily close to each other, even though they do not intersect. In this case we could relocate the focus disk to obtain a more balanced spacing between the labels.

Note that maximizing the smallest angle may be in conflict with minimizing the size of the focus disk.

Experiment 1. For $n = 5, \ldots, 20$, we selected a sample of n random points under normal distribution and computed the radius of the disk maximizing the smallest angle between leaders and covering all sites—the *smallest-angle focus disk*—and the radius of the smallest enclosing disk. For each instance size, we repeated the experiment 50 times. The result was that the radius for the smallest-angle focus disk was on average only 30% larger than the radius of the smallest enclosing disk, with a standard deviation about 20% (see Figure 4(a)).

Using the smallest enclosing disk as focus disk can result in arbitrarily bad angles. To see this, note that we can add a site near the disk center that introduces a small angle between leaders, but will not affect the minimum enclosing disk. Another heuristic for locating the focus disk is the center of gravity. This is motivated by distributing the sites inside the focus disk evenly. We can, however, add two sites that preserve the center of gravity but introduce arbitrarily small angles.

Experiment 2. We repeated Experiment 1, but this time we computed the smallest angle between leaders of the smallest-angle focus disk, the smallest enclosing disk, and the center of gravity disk (see Figure 4(b)). We observed that on average the angle of the smallest-angle disk is twice as large as the angle of the two other disks. This ratio gets even larger for larger point sets.

We repeated Experiments 1 and 2 with uniformly distributed point sets and skewed point sets in which all points lie close to a straight-line segment (that is, we defined the x-coordinates to be uniformly distributed in a small interval and the y-coordinates to be normally distributed). The results (again 50 samples for $n = 5, \ldots, 20$) were very similar to what we recorded in Experiments 1 and 2.

Due to our theoretical considerations and the outcome of our experiments we are convinced that the computationally harder sector-maximization technique (see Sect. 4.4) is worth being applied when selecting the focus disk. Fig. 5 shows a point set with focus disks obtained by the three strategies we just discussed.

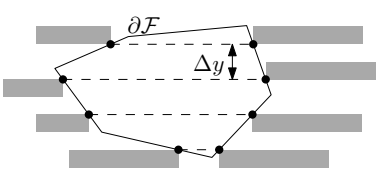


Fig. 6. Construction of port locations (black dots) for a convex focus region $\mathcal{F}$ based on a set of horizontal lines (dashed) with vertical spacing Δy. The gray rectangles represent labels.

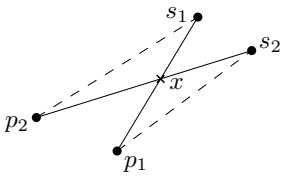


Fig. 7. A crossing of two leaders (solid edges) can always be resolved (dashed edges) while reducing the total leader length.

3.2 Free Leaders

In the free-leader model we do not require that the focus region is a disk, but we assume that it is a (connected) convex region $\mathcal{F}$. As in the radial-leader model, the focus region contains the sites that we want to label and the leaders are straight-line segments, each of which connects a site with a port on $\partial\mathcal{F}$. The locations of the ports on $\partial\mathcal{F}$ may be given as a set P. Alternatively, in case P is not given, we suggest to compute the port locations as shown in Fig. 6.

The method applied in Fig. 6 requires a vertical distance $\Delta y \in \mathbb{R}^+$ between two consecutive ports on $\mathcal{F}$ as input, that is, the spacing between two lines of text. Any intersection between a line and the boundary $\partial\mathcal{F}$ of the focus region defines the location of a label port in the set P. Usually, we define the set of lines such that no line intersects the uppermost or lowermost point of $\mathcal{F}$—due to their prominent positions, labels placed at such points would be perceived to dominate other labels, which we normally want to avoid. In any case, setting Δy to a value greater than the height of a label and using the classical four-position point-labeling model [6] that allows any corner of a label to coincide with the label's port, we can always place the labels such that they intersect neither each other nor the interior of $\mathcal{F}$. As the focus region is convex and contains all sites in its interior, every straight-line segment that connects a port with a site lies completely in $\mathcal{F}$. Hence, intersections between leaders and labels (apart from label ports) are impossible. We have to ensure, however, that no two leaders cross.

We first consider the case that the number of sites and the number of locations for label ports are equal, that is, $|S| = |P|$. In this case, the core question is which sites are assigned to which ports. This can be formalized using graph-theoretic terms. We define the graph $G = (S \cup P, S \times P)$ that contains a vertex for each site, a vertex for each port location, and an edge for each pair of a site and a port location. Then a labeling is defined by a *perfect matching* in G, that is, a subset M of $S \times P$ containing exactly one edge incident to each vertex; each edge $\{s, p\}$ in M defines a leader between a site s and a port p.

Not all perfect matchings in G yield equally good labelings; some may actually imply crossing leaders. To reduce visual clutter, we prefer short leaders. Let $d(s, p)$ be the Euclidean distance of points s and p in the plane. We minimize the sum of this distance over all matched pairs of a site and a port location.

Problem 5. *Minimum distance port assignment*

Input: *Convex region $\mathcal{F}$, set $S \subset \mathcal{F}$ of n sites, set $P \subset \partial\mathcal{F}$ of n ports*

Output: *Perfect matching M in the graph $G = (S \cup P, S \times P)$ such that $\sum_{\{s,p\}\in M} d(s,p)$ is minimized and no two leaders defined by edges in M cross.*

To simplify our discussion, we observe that the explicit requirement for non-crossing leaders can be dropped. Even without that requirement, every optimal solution to Problem 5 is crossing-free as Bekos et al. [6] observed. For a sketch, see Fig. 7.

Observation 1 ([6])**.** *Every perfect matching M in $G = (S \cup P, S \times P)$ that minimizes $\sum_{\{s,p\}\in M} d(s,p)$ defines a crossing-free labeling.*

Due to Observation 1, Problem 5 reduces to the problem of finding a minimum-weight perfect matching in a bipartite graph whose nodes represent points in $\mathbb{R}^2$ and whose edges have weights representing Euclidean distances; for any $\varepsilon > 0$, this problem can be solved in $O(n^{2+\varepsilon})$ time [19].

As discussed in Sect. 3.1, we often cannot label all sites. Interestingly, if we are given n sites and $k \leq n$ port locations, we can find a crossing-free labeling for *any* subset $S' \subseteq S$ of k sites. This also follows from Observation 1. If the sites are weighted and we aim at maximizing the total weight of all labeled sites, we simply have to select the k sites of largest weights, which can be done in $O(n)$ time [20], and can then apply the algorithm for Problem 5 to these sites. This requires $O(n + k^{2+\varepsilon})$ time in total. If there are multiple sites of the same weight, however, selecting an arbitrary subset S' of maximum weight can result in unnecessarily long leaders. Generally, we may even want to relax our requirement for a maximum-weight set of sites to achieve a labeling with shorter leaders, for example, to avoid that clusters of heavy-weighted sites result in clusters of sites labeled with long leaders. In order to define the trade-off between our preferences for sites of high weights and leaders of short length, we introduce a weight factor $\lambda \in [0,1]$.

Problem 6. *Best trade-off between weight and leader length*

Input: *Convex region $\mathcal{F}$, set $S \subset \mathcal{F}$ of n sites with weights $w\colon S \to \mathbb{R}^+$, set $P \subset \partial\mathcal{F}$ of k ports, number $\lambda \in [0,1]$.*

Output: *Matching M with $|M| = \min\{k,n\}$ in the graph $G = (S \cup P, S \times P)$ that maximizes the objective $\lambda \sum_{\{s,p\}\in M} w(s) - (1-\lambda)\sum_{\{s,p\}\in M} d(s,p)$.*

Note that we do not require anymore that the matching is perfect, thus allow some sites or port locations to remain unmatched. By requiring $|M| = \min\{k,n\}$, however, we ensure that the largest possible number of matches is selected—which is still true if there are less sites than locations for ports.

For $\lambda < 1$ we are sure that, if we reduce the leader length and keep the same set of sites labeled, the objective always increases. Hence, by Observation 1, every optimal solution to Problem 6 is free of crossings. For $\lambda = 1$ we can find an optimal solution without crossings by selecting a set $S' \subseteq S$ of highest weight that contains $\min\{k,n\}$ sites and labeling S' with minimum leader length (by solving Problem 6 with $S := S'$ and $\lambda := 0$).

We can solve Problem 6 by finding a maximum-weight matching in the bipartite graph $G = (S \cup P, S \times P)$ if we define the weight of an edge $\{s,p\}$ in G to be $\lambda \cdot w(s) - (1-\lambda)\cdot d(s,p) + C$, where C is a large constant ensuring that all edge weights are positive. For example, we may set C to the diameter of the focus region. This problem can be solved, in $O(k^3 + n^3)$ time, using the Hungarian method [21].

4 Algorithms for the Radial-Leader Model

4.1 Label Maximization with Given Center

We now present an algorithm for Problem 1, that is, we maximize the number of labels in a radial-leader labeling such that, when seen from center c, every two labeled sites are separated by an angle of at least α.

We first select an arbitrary start node $s_1 \in S$ and sort the sites in S lexicographically according to their angles and distances w.r.t. c. Let $\mathcal{S} = \langle s_1, s_2, \ldots, s_n\rangle$ be the resulting sequence. Then, we define the directed graph $G = (S, E)$ that contains the edge $s_i s_j$ with $i < j$ if the angle $\angle s_i c s_j$ between s_i and s_j in c is at least α. We are looking for a longest *closable* path p in G, that is, a path whose start and end vertex also form an angle of at least α.

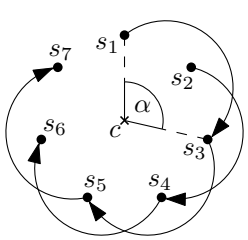


Fig. 9. A reduced graph G'. The longest path in G' is $\langle s_1, s_3, s_5, s_7\rangle$, which is not closable since $\angle s_7 c s_1 < \alpha$. A longest *closable* path is, for example, $\langle s_1, s_3, s_5\rangle$.

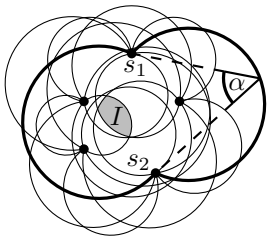


Fig. 10. Arrangement $\mathcal{A}_S^\alpha$ containing the boundary of each partial double disk $U_{u,v}^\alpha$ with $u, v \in S$. Region I is the intersection of all partial double disks. In this example, for every point $c \in I$, (c, S) is an optimum solution to Problem 3.

Let p_{OPT} be a longest closable path in G. Assume that, for two consecutive sites s, s_j in p_{OPT}, there is a site s_i with $i < j$ and $(s, s_i) \in E$. In this case, we can replace s_j with s_i and obtain another longest closable path. Therefore, we can always assume that for every site s in p_{OPT} (except for the last site) the next site t in $\mathcal{S}$ with $(s,t) \in E$ is also contained in p_{OPT}. This allows us to remove many edges from G while still having the guarantee that a longest closable path yields an optimal solution. More precisely, we compute the longest closable path in the reduced graph $G' = (S, E')$, see Fig. 9. The set $E' \subseteq E$ contains at most one outgoing edge for each node $s \in S$, that is, the edge $(s, s_i) \in E$ with smallest index i (if such an edge exists). Clearly, the complexity of G' is $O(n)$ and, if we have already computed the sequence $\mathcal{S}$, we can compute G' in $O(n)$ time.

Choosing a dynamic-programming approach, we now compute for $i = n$ down to 1 the length κ_i and the end σ_i of a longest (not necessarily closable) path in G' that starts in s_i. Obviously, for $i = n$, this is the trivial path containing only s_n, which implies $\kappa_n = 1$ and $\sigma_n = s_n$. For $i < n$, we compute κ_i and σ_i based on the values computed before as $\kappa_i = \kappa_j + 1$ and $\sigma_i = \sigma_j$, where s_j is the successor of s_i in G', that is, the node s_j with $(s_i, s_j) \in E'$. Hence, computing κ_i and σ_i for $1 \leq i \leq n$ takes $O(n)$ time in total.

For each of the paths computed with the dynamic program (that is, for $1 \leq i \leq n$) we can test in $O(1)$ time whether it is closable, simply by testing whether or not $\angle \sigma_i c s_i \geq \alpha$ holds. If the longest path p starting at node s_i is closable, then it is obviously longest among all closable paths starting at s_i. On the other hand, if p is *not* closable we can be sure that (1) there is no closable path of length κ_i starting at s_i and (2) we can obtain a closable path of length $\kappa_i - 1$ by removing the last node from p. In any case, we obtain a longest closable path starting at s_i, which we denote by p^i_{OPT}. Obviously, a path that is longest among $p^1_{\text{OPT}}, p^2_{\text{OPT}}, \ldots, p^n_{\text{OPT}}$ is a longest closable path in G'.

Summing up, it takes us $O(n)$ time to compute a longest closable path, which is dominated by the time needed for sorting S.

4.2 Weighted Label Maximization with Given Center

We consider now the weighted version of the previous problem, that is, Problem 2. In order to find a set S' of maximum weight, we again use the circular order of sites around c, and the graph G' defined in Sect. 4.1. We choose some starting site $s' \in S$, and suppose that $s' \in S'$. Then, we go through all sites in counter-clockwise order, starting at s'. During this process we store for each $s \in S$ the maximum weight $T[s]$ of a conflict-free set in the range from s to and including s'. We compute $T[s]$ as follows.

Let t be the successor of s in G', and let $\tilde{s}$ be the successor of s in clockwise order. Then $T[s] = \max\{w(s) + T[t], T[\tilde{s}]\}$. Hence, we can find a maximum-weight set S' with $s' \in S'$ in $O(n)$ time. By

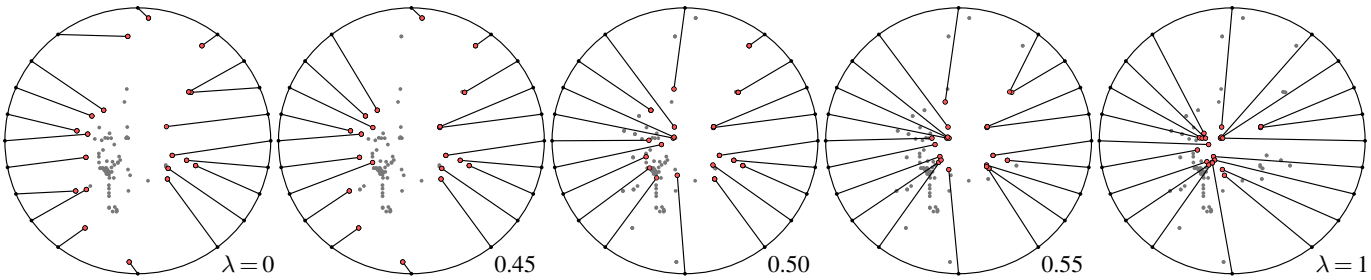


Fig. 8. Solutions to Problem 6 obtained with different values for λ. Since sites close to the center c of the focus region are assumed to be important, the weight $w(s)$ of a site s was set to the Euclidean distance between s and c. Gray dots show sites that did not become labeled.

starting this algorithm at each site s', we maximize the total weight of visible labels in $O(n^2)$ time.

4.3 Label Maximization with Variable Center Position

We now present an algorithm for Problem 3, that is, we maximize the number of non-conflicting sites but now, the center c of the (circular) focus region is not prescribed.

For each pair of sites $s_1, s_2 \in S$ the points from which the line segment $\overline{s_1 s_2}$ appears at an angle of at least α form a region $U^{\alpha}_{s_1,s_2} = \{c \in \mathbb{R}^2 \mid \angle s_1 c s_2 \geq \alpha\}$. According to the *inscribed angle theorem*, the angle at vertex c of a triangle abc does not change if c moves on the circumcircle of the triangle (while staying on the same side of the straight line supported by a and b). Therefore, $U^{\alpha}_{s_1,s_2}$ is the union of two partial disks, one on each side of $\overline{s_1 s_2}$. We thus term $U^{\alpha}_{s_1,s_2}$ a *partial double disk*.

We denote with $\mathcal{A}^{\alpha}_S$ the arrangement given by the union of (the boundaries) of the partial double disks for all pairs of sites, see Fig 10. Since two partial double disks can only intersect $O(1)$ times, the arrangement has a combinatorial complexity of $O(n^4)$. We can traverse the whole arrangement and visit every cell in time $O(n^4 \log n)$ using a sweep line algorithm. During the sweep we determine for each cell C a set $S' \subseteq S$ of maximum size such that for every point $c \in C$ the sites in S' are separated by an angle of at least α (when seen from c). With our algorithm from Sect. 4.1 we can do this in $O(n \log n)$ time. Together with the time needed for traversing all cells we get an $O(n^5 \log n)$-time algorithm for finding an optimal center.

We can reduce the running time of the algorithm to $O(n^5)$ by updating the circular order from cell to cell instead of sorting the points $O(n^4)$ times. The details of this algorithm can be found in Appendix A; see the supplementary material. In the weighted setting, we can combine the traversal of the disk arrangement with the technique presented in Sect. 4.2. This yields an $O(n^6)$-time algorithm that finds an optimum center position.

4.4 Sector Maximization

Let us first address the decision problem derived from Problem 4, that is, we want to decide if we can find a center such that all sites are separated by a given angle α. This question can be answered with the help of the arrangement $\mathcal{A}^{\alpha}_S$ introduced in the previous section. We can find such a center c, if there is a cell in $\mathcal{A}^{\alpha}_S$, that is covered by all possible partial double disks. In order to find such a cell, or to report that none exists, we use again a sweep line algorithm. We keep track of the number of partial double disks that cover the cells we are observing during the sweep. When traversing from one cell to another we can update this information in $O(1)$ time. Hence the decision problem can be solved in $O(n^4 \log n)$ time.

In order to maximize α such that a center c_α with minimum angle at least α exists we use the following technique: Consider the arrangement $\mathcal{A}^{\alpha'}_S$ parametrized by α'. As long as there exists a cell with positive volume that is covered by all partial double disks, we can decrease the angle α' and obtain a new arrangement with such a cell. It follows that for an optimal angle α, there has to be at last one degenerated cell, formed by a single point, that is covered by all partial disks. This however means that three or two of the partial double disks meet in one point. Hence, it suffices to compute for all triplets and tupels of input point pairs the angle, where their induced three (resp., two) partial double disks meet in a single point. The smallest such angle determines the value α. The running time of this strategy is $O(n^6)$.

For Experiment 1 and 2 we have used a prototypical implementation to solve the sector maximization problem. We did not use the proposed (exact) $O(n^6)$ solution, but solved the sector maximization problem numerically. More precisely, we formulated Problem 4 as minimax problem and solved it by the computer algebra software *Mathematica* via the *differential evolution* method. Our solutions converged for almost all computed instances within reasonable time (see Table 1).

Table 1. Runtime for the numerical solution of the sector maximization problem on a standard 2-core 3 GHz desktop computer.

#sites	5	15	25	35	45	55	65	75	85
time[s]	1.83	2.10	2.68	3.82	5.26	6.80	10.07	12.59	15.38

5 Extensions

In this section we discuss two extensions that can be applied to both our leader models. First, we show how to simultaneously cluster sites and label a representative site from each cluster (Sect. 5.1). Second, we transform, in a post-processing step, our straight-line leaders into well-shaped Bézier curves. For the radial-leader model, this gives us more flexibility in the placement of the ports as compared to straight-line leaders. Hence, we can either place more or larger labels. For the free-leader model, Bézier curves help to lead the user's eye in a natural, kink-free way from the site to the label (or vice versa).

5.1 Simultaneous Clustering and Labeling

When discussing the free-leader model (Sect. 3.2) we dealt with the problem that the input map may contain significantly more sites than we have leader ports available. In this case, we had to *select* a subset of sites that are actually labeled and to *assign* these sites to the ports.

In Problem 6, we addressed both issues (selection and assignment) by an extension of bipartite matching. Our experiments indicate, however, that the sites selected by optimum solutions to Problem 6 do *not* sufficiently reflect the spatial distribution of the sites. Consider Fig. 8 that shows various solutions for the same input but different values of the parameter λ. We used a *non-uniform* weight function w where the weight of a site equals its distance to the center of the circle. This can be justified by the assumption that sites closer to the center of the focus region could be considered more important for the user. Simultaneously, this avoids that only sites close to the boundary are labeled, which happens if we use a uniform weight function or if we set $\lambda = 0$ (see solution (a)). Interestingly, this preference of sites close to the boundary persists even if we increase λ to values slightly below 0.5 (see solution (b)). As soon as we increase λ to values greater than or equal to 0.5 (see solutions (c)–(e)) the optimum solution exhibits a preference for sites closer to the center. What we actually want, however, is that the labeled sites are *evenly* distributed thereby reflecting

their spatial distribution. This is not sufficiently accomplished by the solutions for Problem 6 shown in Fig. 8.

The problem of selecting a subset of given cardinality k from a set of input points so that the selected points "represent" the entire set in a spatial sense is a common problem that arises, for example, in *clustering* and *location theory*. Popular formulations as an optimization problem are the k-*means* and the (Euclidean) k-*median* problem. For the k-median problem the input is a set S of points (sites) and a number $k \leq |S|$. The goal is to find a k-element set of *facilities* and to connect each site to a facility so that the *total connection cost*, that is, the sum of the Euclidean distances from the sites to their respective facilities is minimized. In clustering theory, the facilities opened by an optimum solution are considered as *cluster centers* that reflect the spatial distribution of the set S and the clusters are the subsets of sites that are connected to the same cluster center (facility).

In what follows we present a capacitated extension of the Euclidean k-median problem and the closely related facility location problem that incorporates selection and assignment into one neatly formulated optimization problem. We aim at selecting k sites for labeling (considered as facilities) and to connect the remaining sites in S (considered as customers) to a facility. Moreover, we need to ensure that each facility (labeled site) is connected to a *port*. This fits nicely into our location model as we can simply consider the ports as additional, special customers that need to be connected to the facilities. We have to require, however, that each facility serves exactly one port in order to ensure a one-to-one correspondence between labeled sites and ports; see Problem 7. We further assign to each site s a cost $c(s)$ that is incurred if a facility is opened at s. This cost reflects the importance of labeling the site s where a smaller opening cost means higher importance. The overall goal is to minimize the sum of opening costs and total connection cost. We impose an additional *capacity constraint* on each facility that ensures that each facility is connected to roughly the same number of sites. The reason is that in the *uncapacitated* variant optimum solutions tend to place a comparatively small number of facilities into a spatially dense accumulation of points since all the sites in such an accumulation can efficiently be served by few facilities. By adding the capacity constraint we ensure that dense accumulations lead also to accumulations in the set of facilities.

Problem 7. *Facility location based labeling*

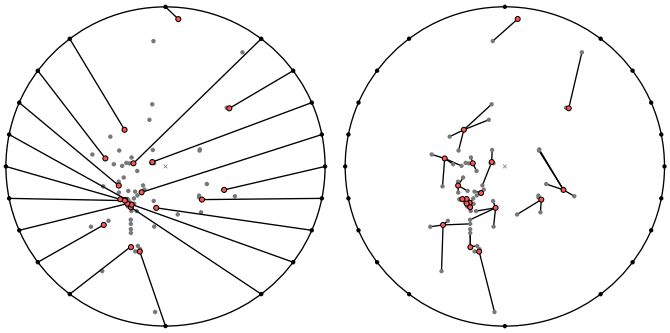
Input: *Convex region $\mathcal{F}$, set $S \subset \mathcal{F}$ of n sites with opening costs $c\colon S \to \mathbb{R}_0^+$, set $P \subset \partial\mathcal{F}$ of $k \leq n$ ports, number $\lambda \in [0,1]$.*

Output: *A k-element set $S' \subseteq S$ of* facilities *and an assignment $\sigma\colon S \cup P \to S'$ specifying for each site or port to which facility it is connected. Each facility must be connected to exactly one port and to at most $\lceil |S|/k \rceil$ sites. A feasible solution has a* total opening cost *of $\sum_{f \in S'} c(f)$ and a* total connection cost *of $\lambda \sum_{p \in P} d(p, \sigma(p)) + (1-\lambda) \sum_{s \in S} d(s, \sigma(s))$. The output is a feasible solution that minimizes the sum of total opening and connection cost.*

Another nice property of our model is that in optimum solutions there are no intersections between port-facility connections and no intersections between site-facility connections. This can be shown analogously to Observation 1. See also Figure 11.

Experiment 3. We formulated Problem 7 as an integer linear program—see Appendix B for details—and implemented it in C++ using the optimizer Gurobi (version 4.6.1). We computed an optimum solution for the same dataset we used in Fig. 8 for discussing Problem 6. It consists of $n = 95$ sites (restaurants on a map excerpt of Seattle) and $k = 20$ ports. The result is shown in Fig. 11. Observe that the selection of labeled sites much better represents the spatial distribution of all sites than the solutions to Problem 6. The computation of the optimum took 124s on a PC with an Intel Core2Duo E8400 CPU with 2 cores at 3 GHz each and 4 GB RAM. In comparison, the computation of the optimum solutions to Problem 6 with 50 different values of α took less than 1s in total on the same PC.

As the integer programming solution is not suitable for interactive usage, we also tested an approach for finding good solutions quickly



(a) optimum solution (b) site-facility connections

Fig. 11. Optimum solution to Problem 7 for Seattle instance ($n = 95$).

in two steps. First, we use a randomized algorithm algorithm for finding k cluster centers. The algorithm chooses new centers one after the other, where, in each step, the probability of choosing a point is proportional to its distance to the closest center chosen so far. This algorithm is known to find a solution to the k-*median* problem with an expected approximation ratio of $O(\log k)$ [22]. Next, we compute crossing-free leaders connecting the chosen centers to the ports by using our matching approach . The algorithm for finding the cluster centers is even faster than the matching algorithms, and gives solutions in which labeled points tend to be in dense regions, but not too close to each other. If a complete clustering is needed, the unlabeled points may be assigned to their closest center; see also Section 5.1.2.

5.1.1 Disk-in-Disk Visualization for Labeling Large Point Sets

When dealing with large and dense point sets, only a small fraction of the sites can actually be labeled since labels must be large enough and may not overlap in order to be readable. We present a natural method based on our facility location model that *clusters* the sites into connected, non-overlapping subregions; one for each label. The user can select a subregion—for example, by clicking the corresponding label—which is then scaled up and labeled in more detail; see Fig. 1. In the new labeling, we keep the selected label fixed, even if this causes crossings with leaders of other labels. We think that this is less distracting than if the selected label jumped to its new position.

We associate each facility f with the set C_f of all sites that are connected to that facility. This set C_f forms a cluster, and the label to C_f is the unique label L_f that is connected to facility f. Note that no cluster contains more than $\lceil |S|/k \rceil$ sites due to the capacity constraints. Our goal is to embed each cluster C_f into a connected subregion $R_f \subseteq \mathcal{F}$ so that no two subregions overlap. A possible use case is to visually highlight the region R_f whenever the cursor hoovers over the label L_f. We want to identify subregions that can be easily recognized by the user. Unfortunately, it is not always possible to find *convex* subregions that meet the above requirements; see Appendix C.

We now propose a simple approach that partitions $\mathcal{F}$ into a collection of subregions containing one cluster each. Our approach is similar to the computation of skeletons based on constrained Delaunay triangulations [23, 24]. Therefore, we defer the details to Appendix D and only provide a sketch here. We define the *star* S_f for facility f as the set of line segments $\overline{fs}$ with $s \in C_f$. Note that no two stars intersect since site-facility connections do not cross. We assume that the boundary $\partial\mathcal{F}$ of $\mathcal{F}$ is given as a polygon. (Otherwise, we can choose a set $\mathcal{F}'$ of sufficiently many points from $\partial\mathcal{F}$ and replace $\mathcal{F}$ with the convex hull of $\mathcal{F}'$.) We compute the union of $\partial\mathcal{F}$ with all stars S_f, $f \in S'$ and complete the resulting set of non-intersecting line segments to a complete triangulation T. For example, we can compute a constrained Delaunay triangulation in $O(n \log n)$ time [25]. We then partition the region $\mathcal{F}$ into connected subregions R_f for each $f \in S'$ such that C_f is contained in R_f. This is accomplished by partitioning each triangle separately as shown in Figure 15(a). Our algorithm ensures that each

site of a cluster C_f is contained in R_f and that two distinct regions $R_f, R_{f'}$ are interior-disjoint. Every region R_f is connected and its boundary ∂R_f is a simple polygon that does not contain sites.

5.1.2 Uncapacitated Relaxation

An interesting relaxation of Problem 7 is to drop the capacity constraints for sites, that is, allowing that an arbitrary number of sites is connected to the same facility. (Of course, we still insist on exactly one port per facility.) This relaxation, gives less incentive to place multiple facilities into spatially dense accumulations of sites since all sites in such an accumulation can efficiently be served by one facility.

A nice property of this relaxed version is that we now can determine a natural decomposition of $\mathcal{F}$ into *convex* and pairwise disjoint regions that cover the clusters of an optimum solution. To this end, consider an optimum solution to the relaxed problem with a set S' of facilities. Now consider the *Voronoi diagram* for the point set S'. Then each facility f lies in a unique Voronoi cell V_f. These cells form a natural decomposition of $\mathcal{F}$ into interior-disjoint *convex* regions. It is easy to see that for each f the cluster C_f is contained in the Voronoi cell V_f.

5.2 Bézier Post-Processing

In all our previous methods, we used crossing-free straight-line leaders for connecting sites and labels. Now we want to improve these drawings by using more flexible leaders. In previous works on boundary labeling, the only drawing styles for leaders have been straight-line segments and polylines [13, Table 1]. We, however, use Bézier curves, which—as curves without sharp bends—are easier to follow than polylines. More precisely, we use cubic Bézier curves, which are defined by their endpoints and two intermediate control points. The survey of Böhm et al. [26] treats these curves in detail.

For computing the new leader layout, we use the *force-directed approach* [27], a well-known general-purpose graph-drawing method. Our algorithms start with a straight-line drawing and let forces, defined by physical analogies, iteratively transform the drawing. We define several forces that are applied to the Bézier curves; each force optimizes a certain aspect of the drawing. In each iteration, the desired movement vectors for all curves are computed by summing up the single forces. Before applying these movements to the current drawing, we limit some movements, if necessary, to ensure that the new drawing is crossing-free. The algorithm terminates when a prespecified number of iterations is reached or the maximum movement per iteration is very small compared to the distances of the input sites.

5.2.1 Horizontal Labels with Given Ports

For the free leader model (Sect. 3.2) our main requirement is that the new leaders enter the labels horizontally from within the focus region. Additionally, the drawing should stay crossing-free, see Problem 8.

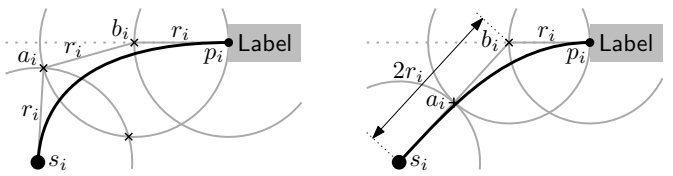
Problem 8. *Bézier leaders for horizontal labels and given ports*

Input: *Disk $D = (c, r)$, sites $s_1, \ldots, s_n \in D$, ports $p_1, \ldots p_n \in \partial D$ such that segments $\overline{s_1 p_1}, \ldots, \overline{s_n p_n}$ are crossing-free.*

Output: *Bézier curves $B_1, \ldots, B_n$ such that, for $i = 1, \ldots, n$, curve B_i connects s_i to p_i and enters port p_i horizontally, and no two Bézier curves intersect. Curves should not be too close to each other and should not have high curvature.*

During our force-directed algorithm, sites and ports will, of course, stay fixed. For $i = 1, \ldots, n$, we initialize curve B_i by making s_i and p_i its endpoints and by placing the two intermediate control points a_i and b_i on their closer endpoint, that is, on s_i and p_i, respectively. Hence, initially $B_i = \overline{s_i p_i}$ and, according to our assumption, this initial drawing is crossing-free. In order to improve the shapes of our "curves", we must move the control points.

It is clear that, if we want the leader to enter port p_i horizontally, b_i must move but stay on the horizontal line through p_i and inside the focus region. We must also move a_i away from s_i to get a good curve. For a nice-looking curve, it makes sense to place the control points so that the three segments of the polyline $s_i a_i b_i p_i$ have roughly equal length r_i. As Fig. 12(a) indicates, there are—under this restriction—two possible positions for a_i if r_i is large enough.



(a) equidistant (b) minimizing equal distance

Fig. 12. Placing control points of a Bézier leader for a horizontal label.

On the other hand, we strive to keep the leader short in order to make it easier to follow. Hence, we try to keep $r_i = d(b_i, p_i)$ as small as possible. In this situation, the only possible position for a_i is the center of $\overline{s_i b_i}$, see Fig. 12(b). To keep visual and computational complexity small, we fix a_i to this position. Our algorithm modifies, therefore, only the parameter $r_i = d(b_i, p_i)$. Let r_i^{OPT} be the optimal value of r_i for B_i, that is, the value resulting in $d(b_i, s_i) = 2r_i^{\text{OPT}}$. Then the attracting force on b_i is $f_{\text{attr}}(B_i) = r_i^{\text{OPT}} - r_i$, that is, the "vector" from the current to the desired position.

An additional criterion to a pleasant-looking leader layout is that two leaders do not come too close. To this end, we add a repelling force between two leaders. Suppose that we have a pair of leaders B_i, B_j, as shown in Fig. 13. We first compute the distance minimum $d(B_i, B_j)$ between the two curves. Note that this can easily be approximated by a polygonization of the curves. Given the relative position of the curves in Fig. 13, the control point b_i of B_i should be repelled towards its port p_i. It is reasonable to make this repelling force larger when the distance between the curves gets smaller. Hence, we set $f_{\text{rep}}(B_i, B_j) = -r_i / d(B_i, B_j)$ for the force repelling B_i from B_j. If the relative position of the curves is different, the force is defined analogously but the direction may change.

Finally, the total force on a control point b_i of curve B_i is $f(B_i) = c_{\text{attr}} f_{\text{attr}}(B_i) + c_{\text{rep}} \sum_{B_j \neq B_i} f_{\text{rep}}(B_i, B_j)$, where the weights $c_{\text{attr}}, c_{\text{rep}}$ are still flexible. In our tests, $c_{\text{attr}} = c_{\text{rep}} = 0.5$ turned out to be a reasonable choice.

Once all forces are computed they should be applied onto the current drawing, that is, the new value of r_i should be $r_i' = r_i + f(b_i)$. Simply applying the forces could, however, lead to crossings, despite the repelling forces between leaders. Therefore we introduce limitations to the forces, that is, a maximum allowed change of the absolute value of r_i. We set this limitation to $f_{\max}(B_i) = 0.45 \min_{B_j \neq B_i} d(B_i, B_j)$. It is easy to see that, with a maximum movement of d on b_i, all points on the new curve B_i' lie within a distance of at most d from the old curve B_i. Hence, by moving b_i and b_j by at most $0.45 d(B_i, B_j)$ we cannot create an intersection of the two leaders. It follows that limiting the absolute value of $f(B_i)$ to $f_{\max}(B_i)$—where necessary—before applying the forces guarantees that the drawing stays crossing-free.

The algorithm terminates when an equilibrium of forces is reached. In practice, it suffices that the maximum change in an iteration is very small, that is, much smaller than the distance between any two sites. Note that it suffices to compute forces just for pairs of leaders that can actually come close. Disregarding all unnecessary pairs gave a huge speedup in our experiments.

Experiment 4. We tested a Java prototype implementation of our algorithm on the same machine used for Experiment 3. We fixed port positions as in Fig. 6. We were mainly interested in the increase of total leader length caused by improving the shape of the leaders. For the test, we used growing subsets of the Seattle instance (see Fig. 1) where all sites had to be labeled (that is, $k = n$). In each new subset, we added sites farther from the city center and increased the radius of the focus region accordingly. For each instance, the algorithm did 200 iterations. Table 2 shows the results.

As we can see, the postprocessing increases the total leader length only by a small percentage while producing nice-looking leaders. For comparison, we refer to the Bézier leader layout in Fig. 1 (center):

Table 2. Runtime for Bézier postprocessing; increase of leader length.

#sites (n)	10	20	30	40	50	60	70	80	90
time [ms]	25	169	123	141	121	170	168	253	354
incr. [%]	4.07	3.09	2.69	2.14	2.57	1.51	1.37	1.90	1.83

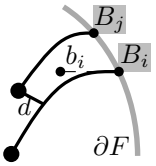


Fig. 13. Curve B_j repells curve B_i.

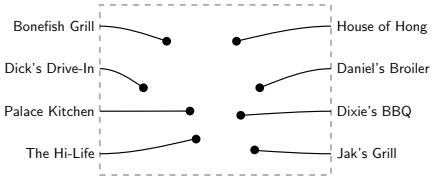


Fig. 14. Rectangular boundary labeling with leaders drawn as Bézier curves.

There, the increase of leader length compared to the straight-line version was only 3.1%, the running time was 0.3s. Note that the running time heavily depends on the distribution of sites. If there are very dense regions such as the cluster in the center, the algorithm is slower due to strong repelling forces which increase the number of necessary iterations. If, however, the sites are better distributed, for example, after selecting a subset with the method presented in Sect. 5.1, the postprocessing is faster and produces better-looking leaders.

5.2.2 Traditional Boundary Labeling

Traditional, two-sided boundary labeling for rectangular maps has, so far, only been approached with straight-line or polyline leaders. It is, however, easy to create such a labeling using Bézier curves as leader using our techniques. Note that we did not use the shape of the focus region in the previous section. Suppose that we are given a rectangular focus region with sites inside and ports on the boundary. Then we can compute a port–label assignment such that straight-line leaders do not cross by the matching technique that solves Problem 5. Using this assignment, we can apply the Bézier postprocessing of the previous section to get a crossing-free layout of Bézier leaders, see Fig. 14.

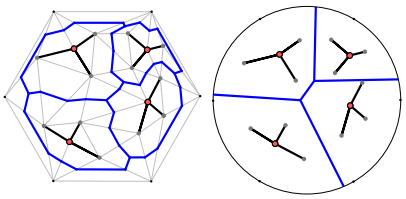
5.2.3 Radial Labels without Given Ports

In the radial-leader scenario we cannot improve much by simply rerouting the leaders using Bézier curves: The leaders already leave sites as well as ports radially and have—as straight-line segments—minimum length. There could, however, be pairs of ports forming a small angle at the center compared to the necessary average angle $360°/n$, even after optimizing the smallest angle with the algorithm presented in Sect. 4.4. We can optimize these angles by moving the ports on the boundary and rerouting the leaders using Bézier curves. We still want the leaders to leave sites and ports radially and insist on crossing-free leaders, see Problem 9.

Problem 9. *Bézier leaders for radial labels without given ports*
Input: *Disk $D = (c, r)$, sites $s_1, \ldots, s_n \in D$.*
Output: *Ports $p_1, \ldots p_n \in \partial D$ and Bézier curves $B_1, \ldots, B_n$ such that the circular order of the ports is the same as the order of their respective sites, for $i = 1, \ldots, n$, curve B_i leaves site s_i radially and enters port p_i radially, and no two Bézier curves intersect. The gaps between the ports should be of approximately equal length and the ports should be close to their sites.*

This problem can again be tackled using a force-directed algorithm: We start by setting, for $i = 1, \ldots, n$, port p_i to the projection of s_i onto ∂D, and by using the straight-line leader $\overline{s_i p_i}$, which is a special Bézier curve. In each iteration, we try to improve the distribution of the ports on ∂D under the additional requirements of Problem 9.

As we want to enter/leave sites and ports radially, the intermediate control points a_i and b_i of the curve B_i connecting site s_i and port p_i must lie on the straight lines cs_i and cp_i, respectively, see



(a) triangulation-based (b) Voronoi-based

Fig. 15. Partitioning the focus region.

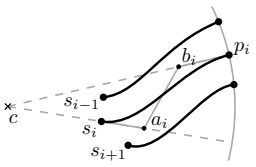


Fig. 16. Bézier curve B_i between neighbors B_{i-1} and B_{i+1}.

Fig. 16. Thus, our algorithm keeps track of three parameters for each Bézier leader B_i: the position of port p_i and the distances $d(s_i, a_i)$ and $d(b_i, p_i)$. We introduce the following forces:

- A force attracting port p_i to the midpoint of $\overline{p_{i-1}p_{i+1}}$. In an equilibrium, that is, if all ports have equal distance to both of their neighbors, all gaps are of equal length.
- A force attracting each port p_i towards its initial position, that is, the radial projection of s_i, for straightening the Bézier curve.
- Repelling forces on the control points if two neighboring curves are too close. This should avoid small distances between the leaders. We try to move both control points to the same direction such that a_i stays closer to the center of the disk than b_i.
- A force that tries to move a_i and b_i such that the distances $d(s_i, a_i)$ and $d(b_i, p_i)$ both are $1/3$ of the distance of s_i to the original position of p_i, which is $r - d(c, s_i)$.

For any force, it is enough to consider the two neighboring sites s_{i-1} and s_{i+1} in circular order. To avoid crossings between labels we limit forces if necessary. This can, again, be done based on the minimum distances to the two neighboring curves.

Experiment 5. The algorithm was implemented in Java and tested on the same machine and the same instances as in Experiment 4. We did 200 iterations per instance. As the positions of ports are flexible, we also measured the optimization criterion, that is, the smallest angle, and compared it to the upper bound $360°/n$. Table 3 shows the results.

Table 3. Runtime, minimum angle (% of $360°/n$), increase of length.

#sites (n)	10	20	30	40	50	60	70	80	90
time [ms]	218	281	324	292	353	400	445	506	574
angle [%]	64.8	26.7	43.3	39.9	48.5	29.3	31.3	37.3	29.2
incr. [%]	11.9	6.4	0.9	1.2	1.0	0.6	0.6	0.4	0.3

For $n \geq 20$, the initial straight-line solution had—due to the dense region of sites in the center—an initial minimum angle of less than $(0.001°)$. With our postprocessing this was always improved to a reasonable percentage of the upper bound $360°/n$. As in Experiment 4, the relative increase of the total leader length was very small. For a visual inspection of the improvement, see Fig. 2(c): when using Bézier curves, the minimum angle increased from $11°$ to $35°$ (upper bound $45°$), whereas the total leader length increased by a mere 6.3%.

6 Conclusion

We have investigated the problem of labeling sites in a focus region by placing labels at the boundary of the focus region and connecting sites and their labels by leaders. We have considered the free-leader model and a new radial-leader model. Usually, users will insist on horizontal labels, which we recommend to combine with the free-leader model. If, however, it is crucial to label a lot of sites (with short labels), it is better to place leaders and labels radially. We strongly recommend to take advantage of the Bézier postprocessing, especially for the radial-leader model which otherwise suffers from small port distances. In a dynamic environment, we suggest to use straight leaders during user interaction. As soon as an interaction ends, one could turn the straight leaders into Bézier leaders by animating the execution of our force-directed algorithm.

REFERENCES

[1] D. Yamamoto, S. Ozeki, and N. Takahashi, "Focus+glue+context: an improved fisheye approach for web map services," in *Proc. 17th Annu. ACM Conf. Advances Geogr. Inform. Syst. (ACM-GIS'09)*. ACM, 2009, pp. 101–110.

[2] J.-H. Haunert and L. Sering, "Drawing road networks with focus regions," *IEEE Trans. Vis. Comput. Graphics*, vol. 17, no. 12, pp. 2555–2562, 2011.

[3] C. Olston and A. Woodruff, "Getting portals to behave," in *Proc. 6th IEEE Symp. Inform. Vis. (InfoVis'00)*, 2000, pp. 15–25.

[4] A. Zipf and K.-F. Richter, "Using focus maps to ease map reading: Developing smart applications for mobile devices," *Künstliche Intelligenz*, vol. 02, no. 4, pp. 35–37, 2002.

[5] M. Steinberger, M. Waldner, M. Streit, A. Lex, and D. Schmalstieg, "Context-preserving visual links," *IEEE Trans. Vis. Comput. Graphics*, vol. 17, no. 12, pp. 2249–2258, 2011.

[6] M. A. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff, "Boundary labeling: Models and efficient algorithms for rectangular maps," *Comput. Geom. Theory Appl.*, vol. 36, no. 3, pp. 215–236, 2007. [Online]. Available: http://dx.doi.org/10.1016/j.comgeo.2006.05.003

[7] M. Wertheimer, "Laws of organization in percetional forms," in *A source book of Gestalt psychology*. Routledge & Kegan Paul, London, 1938, pp. 71–88.

[8] J.-D. Fekete and C. Plaisant, "Excentric labeling: Dynamic neighborhood labeling for data visualization," in *Proc. ACM Conf. Human Factors Comput. Syst. (CHI'99)*, 1999, pp. 512–519.

[9] J. L. Morrison, "Computer technology and cartographic change," in *The Computer in Contemporary Cartography*, D. Taylor, Ed. Johns Hopkins University Press, 1980.

[10] J. Christensen, J. Marks, and S. Shieber, "An empirical study of algorithms for point-feature label placement," *ACM Trans. Graphics*, vol. 14, no. 3, pp. 203–232, 1995. [Online]. Available: http://doi.acm.org/10.1145/212332.212334

[11] F. Wagner, A. Wolff, V. Kapoor, and T. Strijk, "Three rules suffice for good label placement," *Algorithmica*, vol. 30, no. 2, pp. 334–349, 2001. [Online]. Available: http://dx.doi.org/10.1007/s00453-001-0009-7

[12] J.-D. Fekete and C. Plaisant, "Excentric labeling: Dynamic neighborhood labeling for data visualization," Dept. Comput. Sci., Univ. of Maryland, Tech. Rep. HCIL-98-09, 1998. [Online]. Available: http://www.cs.umd.edu/local-cgi-bin/hcil/rr.pl?number=98-09

[13] M. A. Bekos, M. Kaufmann, M. Nöllenburg, and A. Symvonis, "Boundary labeling with octilinear leaders," *Algorithmica*, vol. 57, no. 3, pp. 436–461, 2010.

[14] M. A. Bekos, M. Kaufmann, D. Papadopoulos, and A. Symvonis, "Combining traditional map labeling with boundary labeling," in *Proc. 37th Conf. Current Trends Theory Practice Comput. Sci. (SOFSEM'11)*, ser. Lect. Notes Comput. Sci., I. Cerná, T. Gyimóthy, J. Hromkovic, K. G. Jeffery, R. Královic, M. Vukolic, and S. Wolf, Eds., vol. 6543. Springer-Verlag, 2011, pp. 111–122.

[15] K. Hartmann, T. Götzelmann, K. Ali, and T. Strothotte, "Metrics for functional and aesthetic label layouts," in *Proc. 5th Internat. Symp. Smart Graphics (SG'05)*, ser. Lect. Notes Comput. Sci., A. Butz, B. Fisher, A. Krüger, and P. Olivier, Eds., vol. 3638. Springer-Verlag, 2005, pp. 115–126.

[16] K. Ali, K. Hartmann, and T. Strothotte, "Label layout for interactive 3D illustrations," *J. WSCG*, vol. 13, no. 1, pp. 1–8, 2005.

[17] B. Speckmann and K. Verbeek, "Necklace maps," *IEEE Trans. Vis. Comput. Graphics*, vol. 16, no. 6, pp. 881–889, 2010.

[18] D. Wigdor and R. Balakrishnan, "Empirical investigation into the effect of orientation on text readability in tabletop displays," in *Proc. 9th Europ. Conf. Computer-Supported Cooperative Work (ECSCW'05)*, 2005, pp. 205–224.

[19] P. K. Agarwal, A. Efrat, and M. Sharir, "Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications," *SIAM J. Comput.*, vol. 29, no. 3, pp. 912–953, 1999.

[20] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan, "Time bounds for selection," *J. Comput. Syst. Sci.*, vol. 7, no. 4, pp. 448–461, 1973. [Online]. Available: http://dx.doi.org/10.1016/S0022-0000(73)80033-9

[21] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.

[22] D. Arthur and S. Vassilvitskii, "`k-means++`: The advantages of careful seeding," in *Proc. 18th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA'07)*, 2007, pp. 1027–1035.

[23] R. Chithambaram, K. Beard, and R. Barrera, "Skeletonizing polygons for map generalization," in *Technical papers, ACSM-ASPRS Convention, Cartography and GIS/LIS*, vol. 2, 1991, pp. 44–54.

[24] M. Bader and R. Weibel, "Detecting and resolving size and proximity conflicts in the generalization of polygonal maps," in *Proc. 18th ICA/ACI Int. Cartogr. Conf. (ICC'97)*, Stockholm, 1997, pp. 1525–1532.

[25] P. L. Chew, "Constrained Delaunay triangulations," *Algorithmica*, vol. 4, pp. 97–108, 1989. [Online]. Available: http://dx.doi.org/10.1007/BF01553881

[26] W. Böhm, G. E. Farin, and J. Kahmann, "A survey of curve and surface methods in CAGD," *Comput. Aided Geom. Design*, vol. 1, no. 1, pp. 1–60, 1984.

[27] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Softw. Pract. Exper.*, vol. 21, no. 11, pp. 1129–1164, 1991.

[28] I. Reinbacher, M. Benkert, M. van Kreveld, J. S. Mitchell, J. Snoeyink, and A. Wolff, "Delineating boundaries for imprecise regions," *Algorithmica*, vol. 50, no. 3, pp. 386–414, 2008. [Online]. Available: http://dx.doi.org/10.1007/s00453-007-9042-5