Removing Node Overlaps Using Multi-sphere Scheme*

Takashi Imamichi[1], Yohei Arahori[1], Jaeseong Gim[1], Seok-Hee Hong[2], and Hiroshi Nagamochi[1]

[1] Department of Applied Mathematics and Physics, Kyoto University
{ima,arahori,jaeseong,nag}@amp.i.kyoto-u.ac.jp
[2] School of Information Technologies, University of Sydney
shhong@it.usyd.edu.au

Abstract. In this paper, we consider the problem of removing overlaps of labels in a given layout by changing locations of some of the overlapping labels, and present a new method for the problem based on a packing approach, called *multi-sphere scheme*. More specifically, we study two *new* variations of the label overlap problem, inspired by real world applications, and provide a solution to each problem. Our new approach is very *flexible* to support various operations such as translation, translation with direction *constraints*, and *rotation*. Further, our method can support labels with *arbitrary shapes* in both 2D and *3D layout* settings. Our extensive experimental results show that our new approach is very effective for removing label overlaps.

1 Introduction

Graph Drawing has been extensively studied over the last twenty years due to its popular application for visualization in VLSI layout, computer networks, software engineering, social networks and bioinformatics. As a result, many algorithms and method are available. Note that most algorithms in Graph Drawing deal with abstract graph layout, where each node is represented as a point. However, in many real world applications, nodes may have labels with different size and shape. For example, some nodes have very long text labels or large images, and they can be represented as boxes or circles as in UML diagrams. Consequently, direct use of layout algorithm for abstract graph often leads to overlapping of nodes (i.e. labels) in the resulting visualization.

In order to visualize graphs with different node sizes, the following three steps approach is used in general: (1) a reasonably good initial layout is created using a graph layout algorithm without considering node size; (2) labels of nodes are added in the layout; (3) the post processing step to remove node overlapping is performed. The problem of removing node overlaps has been well studied

* This research was partially supported by Research Fellowships of the Japan Society for the Promotion of Science for Young Scientists and a Scientific Grant in Aid from the Ministry of Education, Science, Sports and Culture of Japan.

I.G. Tollis and M. Patrignani (Eds.): GD 2008, LNCS 5417, pp. 296–301, 2009.

by the Graph Drawing community. These can be classified into three different approaches: force-directed method [5,6,9,11], Voronoi Diagram method [5,11], and constrained optimization method [4,12]. Further, they differ in their optimization criteria considered. The variations of Force Scan algorithm based on the force-directed method [6,9] preserves *orthogonal ordering*, the top-down and left-right relationship between nodes. Note that the problem of transforming a given layout of a graph with overlapping rectangular nodes into a *minimum area* layout without node overlapping which preserves the orthogonal order is proved as NP-complete [6]. The constrained optimization techniques using a quadratic programming approach minimizes the *total change* of node positions while satisfying non-overlap constraints [4,12]. Note that most of the methods solve the problem of overlap removal of *rectangular* labels with *translation* only.

We present a new method for removing overlap of labels based on *multi-sphere scheme* [8], a general algorithmic framework for solving the problem of *packing* objects both in two and three dimensions. Based on this scheme, each label in a given layout is approximated by a set of circles or spheres and a penalty function of the overlap between two labels is introduced. By minimizing the penalty function using a quasi-Newton method [10], we compute a layout of the set of circles or spheres as an approximate solution to the original problem. Our new approach is very *flexible*, and has the following three advantages over previous work:

1. Our approach can handle labels with *arbitrary shapes*. Note that previous methods can deal with only rectangular labels. However, in our approach, we can treat any non-rectangular-shaped labels by approximating each of them as a set of circles. We can also place given labels inside a specified area with a non-rectangular boundary.
2. Our algorithm can use three types of operation: *translation, translation with direction constraints* (i.e. move along the specified line), and *rotation*. Note that the previous methods deal with only translation.
3. Our method can be used for *both 2D and 3D layouts*. Note that previous study can only deal with 2D layout.

In order to demonstrate our three advantages, we consider two new variations of the label overlap problem, each inspired by real world applications, and design an algorithm for each problem setting. More specifically, we consider following two types of label overlap removal problems.

Problem 1: Rectangular Label with Direction-Constrained Translation
Input: A set of overlapping rectangles, where each rectangle is located on its initial position with a specified direction constraints (i.e. a line segment) in the plane.
Output: A set of new positions of rectangles such that no two rectangles overlap and the change of new positions from the initial positions is small, where the new position of each rectangle is obtained by restricted translation along the specified direction only.

Problem 2: 3D Multi-attribute Label with Translation and Rotation
Input: A set of overlapping spiked sphere (i.e. a sphere with several small cones on its surface), where each spiked sphere is located on its initial position in the 3D space.
Output: A set of new positions of spiked spheres in 3D such that no two spiked spheres overlap and the change of new positions from the initial positions is small, where the new position of each spiked sphere is obtained by both translation and rotation in 3D.

Problem 1 appears in applications such as placing labels of street names in a road map layout [2]. Problem 2 appears in applications such as visualization of network data with multiple attributes in three dimensions. For example, a spiked sphere was used to represent an author of the Information Visualization community, where each sphere represents an author, the size of sphere represents the number of research papers published by the author in the conference proceedings, and the length of each spike attached to the sphere represents special attributes such as the number of papers in specific research area [3]. We implemented our algorithm and evaluated with two different types of data sets. Our extensive experimental results show that our new approach is very fast and effective for removing label overlaps. For the full version of this paper, see [7].

2 Algorithm Based on Multi-sphere Scheme

In the multi-sphere scheme, we first approximate each label by a set of spheres, and then search for positions of all the spheres that minimize an appropriate penalty function. Approximating objects by spheres makes it easy to check collisions of objects and handle rotations of objects by arbitrary angles.

To find a layout of sets of spheres, we formulate *penalized rigid sphere set packing problem* of as an unconstrained optimization program and apply an algorithm RigidQN, which moves the labels simultaneously and modifies the entire layout gradually. Given an initial layout of labels, where the labels are approximated by sphere sets, RigidQN returns a locally optimal layout computed by applying the quasi-Newton method to the penalized rigid sphere set packing problem. Although we do not use an explicit criteria to minimize the total change between the initial and final layouts, RigidQN obtains the final positions of labels are close to the initial positions in most cases because RigidQN moves sphere sets gradually.

We formulate the penalized rigid sphere set packing problem for $\mathbb{R}^d$, which asks to move a collection $\mathcal{O} = \{O_1, \ldots, O_m\}$ of m objects so that no two objects overlap each other. Each object O_i consists of n_i spheres $\{S_{i1}, \ldots, S_{in_i}\}$. Let $\boldsymbol{c}_{ij}$ be the vector that represents the center of spheres S_{ij}, r_{ij} be the radius of S_{ij} and $N = \sum_{i=1}^{m} n_i$.We let $\boldsymbol{r}_i = \sum_{j=1}^{n_i} \boldsymbol{c}_{ij}/n_i$, which represents the center of O_i. For a set S of points, let ∂S be the boundary of S, and $\mathrm{int}(S) = S \setminus \partial S$ be the interior of S. After translating object O by a translation vector $\boldsymbol{v} \in \mathbb{R}^d$, the resulting object is described as $O \oplus \boldsymbol{v} = \{\boldsymbol{x} + \boldsymbol{v} \mid \boldsymbol{x} \in O\}$. The *penetration depth* [1] of two

shapes S and T is defined by $\delta(S,T) = \min\{\|\boldsymbol{x}\| \mid \mathrm{int}(S) \cap (T \oplus \boldsymbol{x}) = \emptyset,\ \boldsymbol{x} \in \mathbb{R}^d\}$, where $\|\cdot\|$ denotes the Euclidean norm. Let $\Lambda_i(\boldsymbol{x}, \boldsymbol{v}) : \mathbb{R}^{d \times \lambda_i} \to \mathbb{R}^d$ $(i = 1, \ldots, m)$ be a motion function that moves a point $\boldsymbol{x} \in \mathbb{R}^d$ by λ_i variables $\boldsymbol{v} \in \mathbb{R}^{\lambda_i}$. For a set of points $S \subseteq \mathbb{R}^d$, let $\Lambda_i(S, \boldsymbol{v}) = \{\Lambda_i(\boldsymbol{x}, \boldsymbol{v}) \mid \boldsymbol{x} \in S\}$. For simplicity, we let $\boldsymbol{c}_{ij}(\boldsymbol{v}) = \Lambda_i(\boldsymbol{c}_{ij}, \boldsymbol{v})$ and $S_{ij}(\boldsymbol{v}) = \Lambda_i(S_{ij}, \boldsymbol{v})$. The penalized rigid sphere set packing problem is formally defined by

$$\begin{aligned} \text{minimize} \quad & F_{\mathrm{pen}}(\boldsymbol{v}) = \sum_{1 \le i < k \le m} \sum_{j=1}^{n_i} \sum_{l=1}^{n_k} f_{ijkl}^{\mathrm{pen}}(\boldsymbol{v}), \\ \text{subject to} \quad & \boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m) \in \mathbb{R}^{\sum_{i=1}^{m} \lambda_i}, \\ & \boldsymbol{v}_i \in \mathbb{R}^{\lambda_i}, \quad i = 1, \ldots, m, \end{aligned} \tag{1}$$

where $f_{ijkl}^{\mathrm{pen}}(\boldsymbol{v}) = [\delta(S_{ij}(\boldsymbol{v}_i), S_{kl}(\boldsymbol{v}_k)]^2$ denotes the penetration penalty of two spheres S_{ij} and S_{kl}. This problem is an unconstrained nonlinear program. If the motion functions $\boldsymbol{c}_{ij}(\boldsymbol{v})$ of the centers of objects are chosen to be differentiable, F_{pen} is also differentiable and we can apply the quasi-Newton method to (1). In this paper, we consider following two types of motions.

Translations with a Fixed Direction in 2D for Problem 1. We first consider the case where object O_i is allowed to translate only in a prescribed direction in $\mathbb{R}^2$, but not allowed to rotate. Assume that the reference point $\boldsymbol{r}_i$ of object O_i lies on a line $\boldsymbol{d}_i + t_i \boldsymbol{e}_i$, where $\boldsymbol{d}_i, \boldsymbol{e}_i \in \mathbb{R}^2$ are given and t_i is a variable. Then

$$\Lambda_i(\boldsymbol{x}, t_i) = \boldsymbol{x} - \boldsymbol{r}_i + \boldsymbol{d}_i + t_i \boldsymbol{e}_i, \qquad \frac{\partial \boldsymbol{c}_{ij}(t_i)}{\partial t_i} = \frac{\partial \Lambda_i(\boldsymbol{c}_{ij}, t_i)}{\partial t_i} = \boldsymbol{e}_i.$$

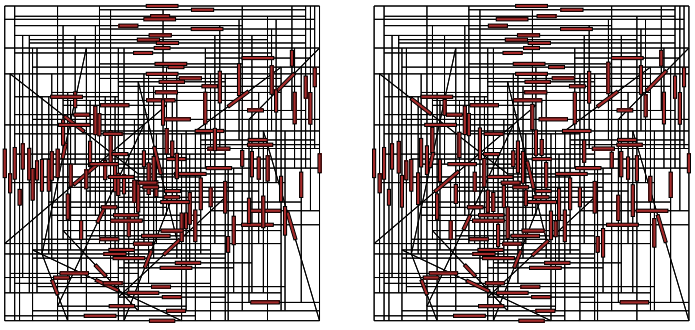
Translations and Rotations in 3D for Problem 2. We next consider the case where each object O_i in $\mathbb{R}^3$ is allowed to translate and rotate around its reference point $\boldsymbol{r}_i$. Let $(x_i, y_i, z_i)^{\mathsf{T}}$ be the translation vector, $(\phi_i, \theta_i, \psi_i)$ be the z-x-z Euler angles, and $R_3(\phi_i, \theta_i, \psi_i)$ be the rotation matrix. Given variables $\boldsymbol{v}_i = (x_i, y_i, z_i, \phi_i, \theta_i, \psi_i)^{\mathsf{T}}$, we define the resulting position of a point $\boldsymbol{x} \in \mathbb{R}^3$ after the motion by $\Lambda_i(\boldsymbol{x}, \boldsymbol{v}_i) = R_3(\phi_i, \theta_i, \psi_i)(\boldsymbol{x} - \boldsymbol{r}_i) + (x_i, y_i, z_i)^{\mathsf{T}} + \boldsymbol{r}_i$. Then,

$$\frac{\partial \boldsymbol{c}_{ij}(\boldsymbol{v}_i)}{\partial x_i} = (1, 0, 0)^{\mathsf{T}}, \qquad \frac{\partial \boldsymbol{c}_{ij}(\boldsymbol{v}_i)}{\partial \phi_i} = \frac{\partial R_3(\phi_i, \theta_i, \psi_i)}{\partial \phi_i}(\boldsymbol{c}_{ij} - \boldsymbol{r}_i).$$

The other derivatives of $\boldsymbol{c}_{ij}(\boldsymbol{v}_i)$ with respective to y_i, z_i, θ_i, and ψ_i can be calculated analogously.

3 Experimental Results

We conducted computational experiments of RigidQN by generating instances of both Problems 1 and 2 randomly. We implemented RigidQN in C++, compiled it by GCC 4.1 and conducted experiments on a PC with an AMD Sempron 3000+ 1.8 GHz processor and 450 MB memory. We adopted a quasi-Newton method package L-BFGS [10].



(a) (b)

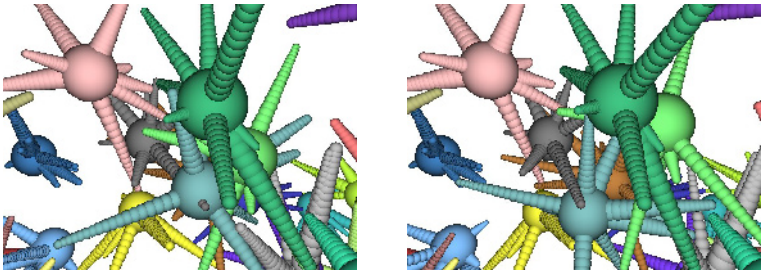
Fig. 1. An example of a road map layout with 147 labels ($\ell_{\text{label}} = 100, \ell_{\text{grid}} = 150$)

The data set of Problem 1 was generated as follows. We first start with a square with size $\ell_{\text{map}} \times \ell_{\text{map}}$ which consists of four lines as a drawing area, where we set $\ell_{\text{map}} = 10000$, and place a square grid on the square, where the minimum distance between two grid lines is ℓ_{grid}. Next we draw horizontal and vertical line segments one after another on the grid lines. Then, we draw some slanted line segments by choosing two arbitrary points in the drawing. Finally, we place a rectangle in the middle of each line segment in the drawing, where the height of a rectangle is ℓ_{label} and the length of a rectangle varies over a range $[5\ell_{\text{label}}, 10\ell_{\text{label}}]$. For example, Fig. 1(a) shows an initial layout with labels for $\ell_{\text{label}} = 100$, where a line segment represents an edge (i.e. street) and a rectangle represents a label (i.e. street name). Figure 1(a) is generated for $\ell_{\text{grid}} = 150$, which has 147 labels and 4818 circles. See Fig. 1(b) for the resulting layout. It took 0.43 seconds for Fig. 1(b).

To observe the influence of the density of the road map layout and the number of labels on the efficiency of our algorithm, we varied two parameters ℓ_{label} and ℓ_{grid} from 100 to 1000 with a step size 50 and from 50 to 1000 with a step size 50, respectively, and conducted experiments. For each setting, we generated 100 instances and applied RigidQN to them. We observed that our algorithm removed almost all overlaps in less than one second for the instances for $\ell_{\text{grid}} \geq 2\ell_{\text{label}}$. For details on the experimental results, see [7].

For the data set of Problem 2, we created instances which resemble the spiked spheres used in [3]. We generated an instance as follows. A spiked sphere has a sphere of radius 10 together with attached 10 spikes. Each spike consists of 20 spheres and the length varies on a range $[10, 70]$. To create an instance, we place the spiked spheres randomly in a cube with edge length 300, where the number of spiked spheres is a parameter. See Fig. 2 for magnified pictures of the initial layout and the resulting layout of an instance with 100 spiked spheres. We can see a spiked sphere in Fig. 2(a) penetrating another spiked sphere, and the removal of overlap in Fig. 2(b). RigidQN run in 1.7 seconds for Fig. 2(b).

To observe the influence of the number of spiked spheres on the efficiency of our algorithm, we varied the number of spiked spheres from 50 to 250 with a step



(a) (b)

Fig. 2. Magnified figures of an instance with 100 labels of Problem 2

size 50, generated 10 instances for each setting, and measured the computation time. We observed that our algorithm removed almost all overlaps in less than 10 seconds for the instances with the number of spikes spheres less than or equal to 200. For details on the experimental results, see [7].

References

1. Agarwal, P.K., Guibas, L.J., Har-Peled, S., Rabinovitch, A., Sharir, M.: Penetration depth of two convex polytopes in 3D. Nordic Journal of Computing 7(3), 227–240 (2000)
2. Agrawala, M.: Visualizing Route Maps. Ph.D. thesis, Stanford University (2002)
3. Ahmed, A., Dwyer, T., Hong, S.H., Murray, C., Song, L., Wu, Y.X.: Visualisation and analysis of large and complex scale-free networks. In: EUROVIS 2005: Eurographics / IEEE VGTC Symposium on Visualization, pp. 239–246 (2005)
4. Dwyer, T., Marriott, K., Stuckey, P.J.: Fast node overlap removal. In: Healy, P., Nikolov, N.S. (eds.) GD 2005. LNCS, vol. 3843, pp. 153–164. Springer, Heidelberg (2006)
5. Gansner, E.R., North, S.C.: Improved force-directed layouts. In: Whitesides, S.H. (ed.) GD 1998. LNCS, vol. 1547, pp. 364–373. Springer, Heidelberg (1999)
6. Hayashi, K., Inoue, M., Masuzawa, T., Fujiwara, H.: A layout adjustment problem for disjoint rectangles preserving orthogonal order. Systems and Computers in Japan 33(2), 31–42 (2002)
7. Imamichi, T., Arahori, Y., Gim, J., Hong, S.H., Nagamochi, H.: Removing overlaps in label layouts using multi-sphere scheme. Tech. Rep. 2008-006, Dept. of Applied Mathematics and Physics, Kyoto University (2008)
8. Imamichi, T., Nagamochi, H.: A multi-sphere scheme for 2D and 3D packing problems. In: Stützle, T., Birattari, M., Hoos, H.H. (eds.) SLS 2007. LNCS, vol. 4638, pp. 207–211. Springer, Heidelberg (2007)
9. Li, W., Eades, P., Nikolov, N.: Using spring algorithms to remove node overlapping. In: APVis 2005. CRPIT, vol. 45, pp. 131–140 (2005)
10. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Mathematical Programming 45(3), 503–528 (1989)
11. Lyons, K.A., Meijer, H., Rappaport, D.: Algorithms for cluster busting in anchored graph drawing. Journal of Graph Algorithms and Applications 2(1), 1–24 (1998)
12. Marriott, K., Stuckey, P., Tam, V., He, W.: Removing node overlapping in graph layout using constrained optimization. Constraints 8(2), 143–171 (2003)