Removing Overlaps in Label Layouts 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

Technical report 2008-006, June 4, 2008.

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*. Based on this scheme, each label in a given layout is approximated by a set of circles, and a cost function that penalizes the overlap between two objects is introduced. By minimizing the penalty function using a quasi-Newton method, we compute a layout of the set of circles as an approximate solution to the original problem.
We consider 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.

Keywords: Graph Drawing, Node Overlap Removal, Map Labeling, Road Map Layout, Multivariate Network.

1 Introduction

1.1 Motivation and Background

Graph Drawing has been extensively studied over the last twenty years due to its popular application for visualisation in VLSI layout and visualization of computer networks, software engineering, social networks and bioinformatics. As a result, many algorithms and method are available [4].

⋆ 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.

Note that most algorithms and methods 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. First, a reasonably good initial layout is created using a graph layout algorithm without considering node size. Second, labels of nodes are added in the layout. Finally, the post processing step to remove node (i.e. label) overlapping is performed.

1.2 Related Work

The problem of removing node overlaps has been well studied for the last ten years by the Graph Drawing community. These can be classified into three different approaches: methods based on force-directed algorithm [6–8, 10, 12, 14], methods based on the use of Voronoi Diagram [6, 12], and methods using constrained optimization techniques [5, 13].

Further, they differ in their optimization criteria considered. The variations of Force Scan algorithm based on the force-directed method [7, 8, 10, 14] 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 [7]. The constrained optimization techniques using a quadratic programming approach minimizes the *total change* of node positions while satisfying non-overlap constraints [5, 13].

The time complexity is $O(n^2)$ for force-directed methods [6–8, 10, 12, 14], however for some special cases, it can be reduced to $O(n \log n)$ [5, 12].

Note that most of the methods solve the problem of overlap removal of *rectangular* labels with *translation* only.

1.3 Our Contribution

We present a new method for removing overlap of labels based on *multi-sphere scheme* [9], 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, and a cost function that penalizes the overlap between two circles is introduced so that the cost function takes value 0 if the current layout of sets of circles has no overlap. By minimizing the penalty function using a quasi-Newton method, we compute a layout of the set of circles as an approximate solution to the original problem.

Our new approach is very *flexible*, and has the following three advantages over previous work.

1. First, 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. Second, 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. Finally, 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 present an algorithm for removing label overlaps with two different variations:

- **Problem 1:** rectangular labels with direction constrained translation (inspired from road map layout [1]), and
- **Problem 2:** 3D labels of arbitrary shape with both translation and rotation (inspired from 3D visualization with multiple attributes of nodes [2]).

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.

The remainder of the paper is organized as follows. In Section 2, we formally define two new variations of label overlapping problem. Section 3 presents an algorithm for the label layout problem based on multi-sphere scheme. Section 4 presents experimental results for two different real world applications. We conclude with future work in Section 5.

2 Problem Definition

The label overlap removal problem is to remove a set of overlapping labels in the given layout by modifying the positions of the labels, so that no two labels overlap each other.

We design an algorithm for label overlap removal problem based on the multi-sphere scheme [9], a flexible framework for general packing problem, where one can choose arbitrary shapes of objects, and restrict the movement of each object with translation and rotation.

We now formally define two types of label overlap removal problem.

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 (i.e. on the line segment where the label initially lies).

Problem 1 appears in applications such as placing labels of street names in a road map layout [1].

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 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 [2].

Based on each type of label overlap removal problem, we design algorithms for two different problem settings. More specifically, we formulate each problem as an optimization problem by introducing an objective function in order to remove the overlap in a given layout (the formal description will be given in the next Section).

Although we do not use an explicit criteria to minimize the total change between the initial and final layouts, our algorithm repeatedly modifies the initial layout by finding the best direction of translation of each object until a new layout with no overlap is obtained as a local optimal solution to our optimization problem. Thus in most cases, the final positions of labels are close to the initial positions.

3 Algorithm based on Multi-sphere Scheme

The multi-sphere scheme is an approach to design efficient algorithms that compute compact layouts of given objects for the packing problem in 2D and 3D space [9]. In the multi-sphere scheme, we first approximate each object by a set of spheres, and then search for positions of all the spheres that minimize an appropriate penalty function.

For this, we formulate the problem of finding a layout of sets of spheres as an unconstrained optimization problem. This optimization problem can provide us an efficient procedure for modifying a given layout into a new layout with no overlap, where such a layout is obtained as a locally optimal solution to the optimization problem. Approximating objects by spheres makes it easy to check collisions of objects and handle rotations of objects by arbitrary angles.

Note that the multi-sphere scheme is very general and can handle both rigid and deformable objects. In this paper, we only use the rigid case of the multi-sphere scheme.

We first briefly review a local search algorithm, called RigidQN designed for the rigid case [9]. Given an initial layout of objects and a container for packing, RigidQN searches for a layout with no overlap and no protrusion by translating and rotating the objects.

We first 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 $N = \sum_{i=1}^{m} n_i$ be the total number of spheres. 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 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* [3] 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. For spheres S_{ij} and S_{kl}, the penetration depth of them is $\delta(S_{ij}, S_{kl}) = \max\{r_{ij} + r_{kl} - \|\boldsymbol{c}_{ij} - \boldsymbol{c}_{kl}\|, 0\}$.

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 defined by

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

where

$$\begin{aligned} f_{ijkl}^{\text{pen}}(\boldsymbol{v}) &= [\delta(S_{ij}(\boldsymbol{v}_i), S_{kl}(\boldsymbol{v}_k))]^2 \\ &= [\max\{r_{ij} + r_{kl} - \|\boldsymbol{c}_{ij}(\boldsymbol{v}_i) - \boldsymbol{c}_{kl}(\boldsymbol{v}_k)\|, 0\}]^2 \end{aligned}$$

denotes the penetration penalty of two sphere S_{ij} and S_{kl}.

The penalized rigid sphere set packing problem is an unconstrained nonlinear program and the objective function F_{pen} is chosen to be differentiable [9]. If

$S_{ij}(\boldsymbol{v}_i)$ and $S_{kl}(\boldsymbol{v}_k)$ intersect each other, then it holds that

$$\frac{\partial f^{\mathrm{pen}}_{ijkl}(\boldsymbol{v})}{\partial \boldsymbol{v}_i} = -2\delta(S_{ij}(\boldsymbol{v}_i), S_{kl}(\boldsymbol{v}_k)) \cdot \frac{\partial \boldsymbol{c}_{ij}(\boldsymbol{v}_i)^{\mathsf{T}}}{\partial \boldsymbol{v}_i} \cdot \frac{\boldsymbol{c}_{ij}(\boldsymbol{v}_i) - \boldsymbol{c}_{kl}(\boldsymbol{v}_k)}{\|\boldsymbol{c}_{ij}(\boldsymbol{v}_i) - \boldsymbol{c}_{kl}(\boldsymbol{v}_k)\|},$$

where $^{\mathsf{T}}$ denotes the transpose of a vector/matrix. Otherwise, it holds that $\partial f^{\mathrm{pen}}_{ijkl}(\boldsymbol{v})/\partial \boldsymbol{v}_i = \boldsymbol{0}$.

In the multi-sphere scheme, we can impose a restriction on the motion of each object O_i by defining its motion function Λ_i appropriately. In this paper, we consider two different motions of objects to solve the problems listed in the previous section.

We now describe specific details more formally.

3.1 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

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

This formulation can handle Problem 1 by approximating each rectangular label by a set of circles, because Problem 1 allows translation of each label in a fixed direction.

3.2 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

$$\Lambda_i(\boldsymbol{x}, \boldsymbol{v}_i) = R_3(\alpha_{\mathrm{rot}}\phi_i, \alpha_{\mathrm{rot}}\theta_i, \alpha_{\mathrm{rot}}\psi_i)(\boldsymbol{x} - \boldsymbol{r}_i) + (x_i, y_i, z_i)^{\mathsf{T}} + \boldsymbol{r}_i,$$

where a positive parameter α_{rot} denotes sensitivity of rotations (we set $\alpha_{\mathrm{rot}} = 10^{-1}$ in our experiments). Then,

$$\begin{aligned}\frac{\partial \boldsymbol{c}_{ij}(\boldsymbol{v}_i)}{\partial x_i} &= (1, 0, 0)^{\mathsf{T}},\\ \frac{\partial \boldsymbol{c}_{ij}(\boldsymbol{v}_i)}{\partial \phi_i} &= \frac{\partial R_3(\alpha_{\mathrm{rot}}\phi_i, \alpha_{\mathrm{rot}}\theta_i, \alpha_{\mathrm{rot}}\psi_i)}{\partial \phi_i}(\boldsymbol{c}_{ij} - \boldsymbol{r}_i).\end{aligned}$$

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.

This formulation can handle Problem 2 by approximating each “spiked sphere” by a set of spheres, because Problem 2 allows translation and rotations of 3D objects.

3.3 Local Search Algorithm

The local search algorithm RigidQN applied to the label layout problem repeatedly modifies the positions of labels as follows.

Given an initial layout $\mathcal{O}$ of labels, where the labels are approximated by sphere sets, RigidQN($\mathcal{O}$) returns a locally optimal layout computed by applying the quasi-Newton method to the penalized rigid sphere set packing problem (1).

RigidQN moves the labels simultaneously and modifies the entire layout gradually until the value of the penalty function becomes 0 (i.e. the overlap of labels is removed), where the motions of objects are followed by the motion functions $\{\Lambda_1, \ldots, \Lambda_m\}$. In practice, RigidQN is implemented so that the computation halts if the value of the penalty function becomes sufficiently small.

To apply the multi-sphere scheme to Problems 1 and 2, we treat the problem as a packing problem which asks to move given labels in a sufficiently large container. We then approximate each label in a given layout as a set of spheres that covers the interior of the label, and apply the local search algorithm RigidQN .

4 Experimental Results

We conducted computational experiments of RigidQN by generating instances of both Problems 1 and 2 randomly. In this section, we report the results.

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 [11]. L-BFGS has a parameter m_{BFGS}, which is the number of BFGS corrections in L-BFGS. We set $m_{\mathrm{BFGS}} = 6$ as it is recommended to choose from the range $3 \leq m_{\mathrm{BFGS}} \leq 7$ in the L-BFGS package.

4.1 Results of Problem 1

In Problem 1, the movement of rectangular labels is limited to translation in specified directions, as appeared in the road maps. Thus, we generated instances which represent the road map label layout, where the labels (i.e. names of the streets) are constrained to be placed along the corresponding streets.

More specifically, we are given a set of edges embedded in the plane, where each edge is drawn as a straight-line segment that represents a street in the road map, and required to remove overlaps of rectangular labels by moving each label along each edge. Note that two edges (i.e. two streets) may have an intersection in the plane, in general.

The data set was generated as follows. We first start with a square with size $\ell_{\mathrm{map}} \times \ell_{\mathrm{map}}$ which consists of four lines as a drawing area, where we set $\ell_{\mathrm{map}} = 10000$, 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. To draw a horizontal line segment, we choose two arbitrary vertical line segments whose distance is more than or equal to $\ell_{\mathrm{map}}/3$ and connect them with an arbitrary horizontal line segment on the grid (we draw a vertical line segment analogously).

We repeat drawing line segments until we cannot choose any pair of line segments.

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 $[5\ell_{\mathrm{label}}, 10\ell_{\mathrm{label}}]$.

For example, Figures 1(a), 2(a) and 3(a) show instances with initial positions of labels for $\ell_{\mathrm{label}} = 100$, 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 $\ell_{\mathrm{grid}} = 200$, which has 112 labels and 3601 circles. Figure 2(a) is generated for $\ell_{\mathrm{grid}} = 150$, which has 147 labels and 4818 circles. Figure 3(a) is generated for $\ell_{\mathrm{grid}} = 100$, which has 222 labels and 6773 circles.

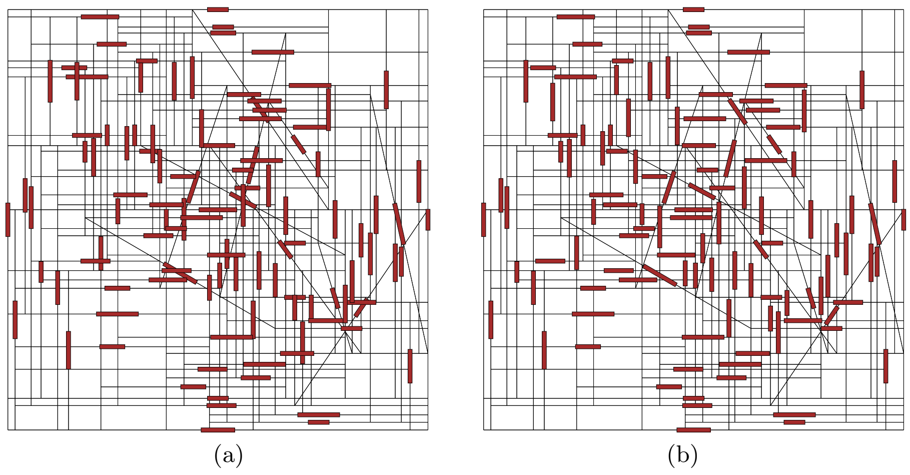


Fig. 1. An example of a road map layout with 112 labels ($\ell_{\mathrm{label}} = 100, \ell_{\mathrm{grid}} = 200$) : (a) an initial layout, (b) a final layout.

Here we consider the problem of removing the overlap between all pairs of rectangles in the initial layout by moving each rectangle along the corresponding line segment. Again, we apply our algorithm RigidQN after approximating each rectangle by a set of circles.

Note that for this approximation, we add more circles around the center of a rectangle. This is because we observed form our preliminary experiment that it was better than approximating a rectangle by placing circles equally. Thus, we place circles by increasing the number of circles from both ends of label as follows.

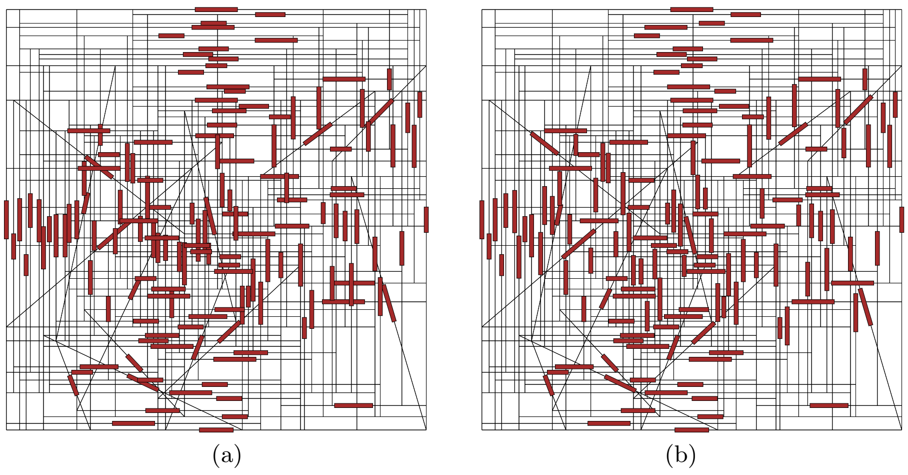


Fig. 2. An example of a road map layout with 147 labels ($\ell_{\text{label}} = 100, \ell_{\text{grid}} = 150$) : (a) an initial layout, (b) a final layout.

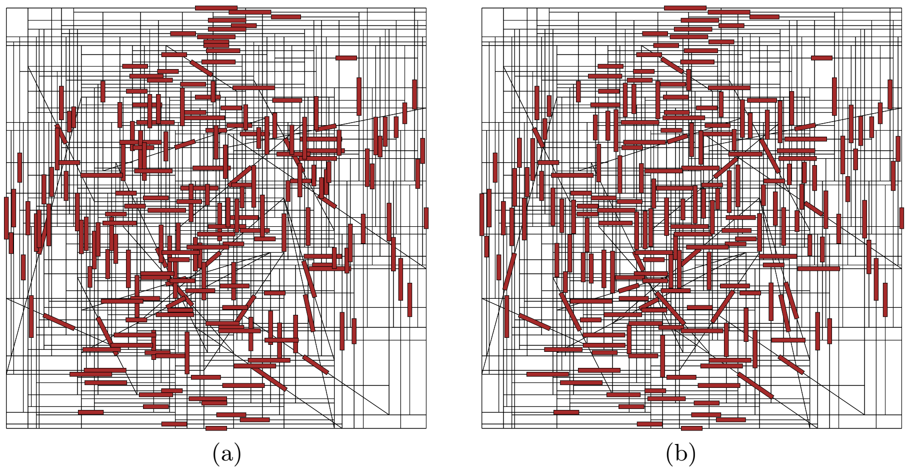


Fig. 3. An example of a road map layout with 222 labels ($\ell_{\text{label}} = 100, \ell_{\text{grid}} = 100$) : (a) an initial layout, (b) a final layout.

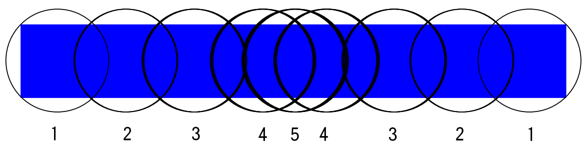


Fig. 4. Approximation of a rectangle for road map instances.

Assume that we are given a rectangle whose width w is longer than the height h and that the bottom left point of the rectangle lies at the origin. We place circles with a radius $h/2$ at

$$\{((i-1/2)h, h/2), (w-(i-1/2)h, h/2) \mid i = 1, \ldots, \lfloor (w/h+1)/2 \rfloor\} \\ \cup \{(w/2, h/2)\}.$$

The numbers of circles to place at $((i-1/2)h, h/2)$ and $(w-(i-1/2)h, h/2)$ are both i and the number of circles to place at $(w/2, h/2)$ is $\lceil (w/h+1)/2 \rceil$. See Figure 4 for an example. In the figure, the number under a circle represents the number of circles on the position.

See Figures 1(b), 2(b) and 3(b) for the final layouts of the instances of Figures 1(a), 2(a) and 3(a) after removing overlaps of labels by our method, respectively. It took 0.24 seconds for Figure 1(b), 0.43 seconds for Figure 2(b) and 1.5 seconds for Figure 3(b). Note that our method successfully removed all the overlaps for the sparse instance. However, for some dense instances, it left a few overlaps.

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 generate 100 instances and apply RigidQN to them.

Table 1 shows details of some of the the selected results. The column "time," "decrease," "#label," and "#circles" denote the average computation time, the average ratio of the decrease of the object function (1), the average number of labels, and the average number of circles, respectively. Figures 5 and 6 show the average computation time and the average ratio of the penalty decrease for $\ell_{\text{label}} = 200$, respectively. We observed that our algorithm removed almost all overlaps in less than one second for the instances for $\ell_{\text{grid}} \geq 2\ell_{\text{label}}$.

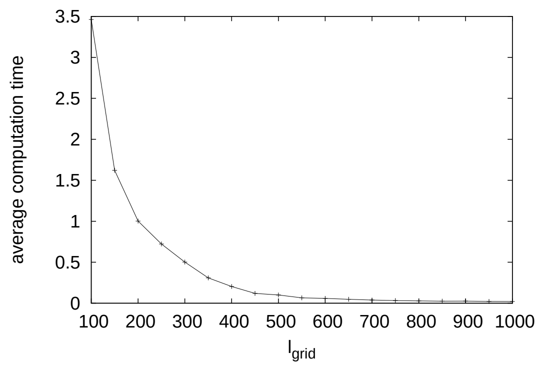


Fig. 5. Average computation time for $\ell_{\text{label}} = 200$.

ℓ_{label}	ℓ_{grid}	time	decrease (%)	#labels	#circles
100	250	0.12	99.9	9.0×10^1	2.8×10^3
	200	0.22	99.6	1.1×10^2	3.4×10^3
	150	0.65	98.8	1.4×10^2	4.5×10^3
	100	1.57	98.3	2.2×10^2	6.9×10^3
200	400	0.20	98.6	5.7×10^1	1.7×10^3
	350	0.30	98.3	6.3×10^1	1.9×10^3
	300	0.50	97.0	7.4×10^1	2.3×10^3
	250	0.72	95.5	9.0×10^1	2.8×10^3
300	600	0.16	97.4	3.7×10^1	1.1×10^3
	500	0.28	96.4	4.6×10^1	1.4×10^3
	400	0.48	92.4	5.7×10^1	1.7×10^3
	300	0.79	83.5	7.4×10^1	2.3×10^3
400	1000	0.07	98.0	2.4×10^1	7.5×10^2
	800	0.12	94.6	2.8×10^1	8.8×10^2
	600	0.29	89.9	3.7×10^1	1.1×10^3
	400	0.66	74.6	5.7×10^1	1.7×10^3
500	1000	0.12	89.4	2.4×10^1	7.5×10^2
	800	0.19	87.4	2.8×10^1	8.7×10^2
	600	0.32	81.1	3.7×10^1	1.1×10^3
	400	0.71	65.5	5.7×10^1	1.8×10^3

Table 1. Details of the selected results of road map instances.

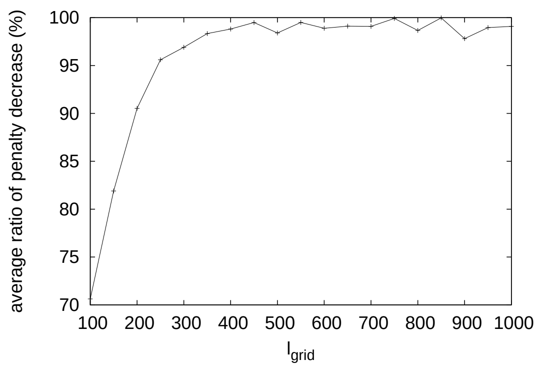


Fig. 6. Average ratio of penalty decrease for $\ell_{\text{label}} = 200$.

4.2 Results of Problem 2

For the data set of Problem 2 (i.e. 3D objects of various shapes with rotation and translation), we create an instance which resembles the spiked spheres used in [2]. More specifically, given a layout of a set of spiked spheres with some overlaps, we remove the overlaps by translation and rotation.

We generate 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 $[10, 70]$. Thus a spiked sphere has 201 spheres in total. 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 Figures 7(a), 8(a) and 9(a) for instances with 50, 100 and 200 spiked spheres, respectively. See Figures 7(b), 8(b) and 9(b) for the resulting layouts. Our algorithm RigidQN run in 0.47 seconds for Figure 7(b), 1.7 seconds for Figure 8(b) and 8.8 seconds for Figure 9(b), and obtained layouts with no overlap of spiked spheres by translating and rotating them slightly. Figures 10(a) and (b) are magnified pictures of Figures 8(a) and (b), respectively. We can see a spiked sphere in Figure 10(a) penetrating another spiked sphere, and the removal of overlap in Figure 10(b).

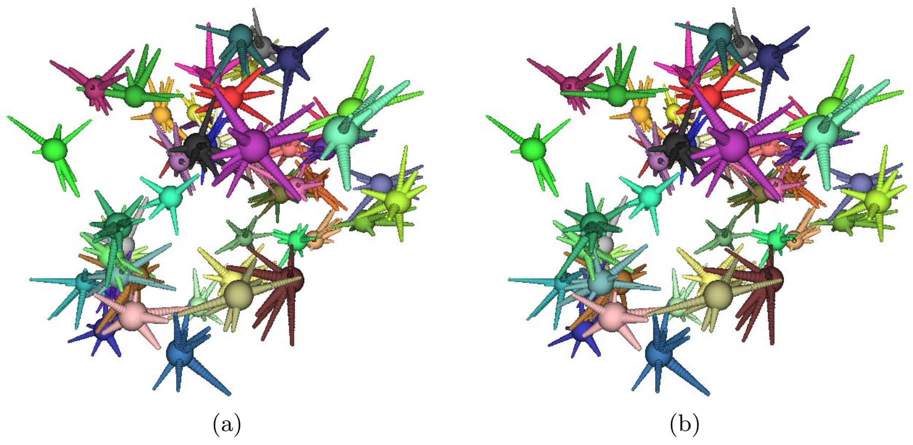
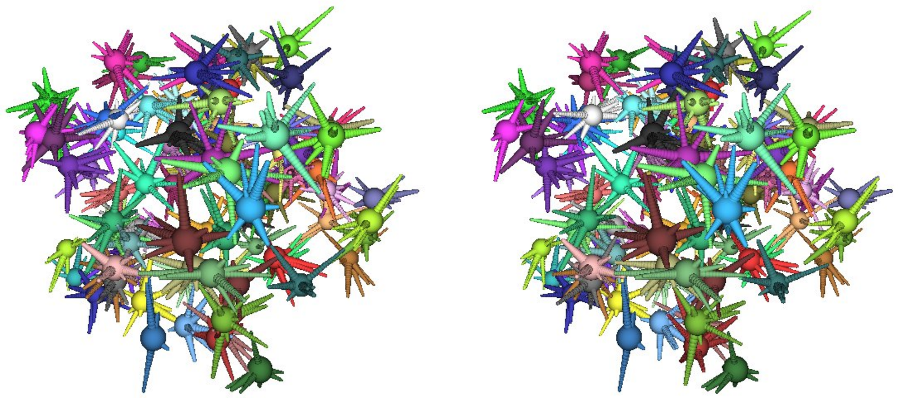


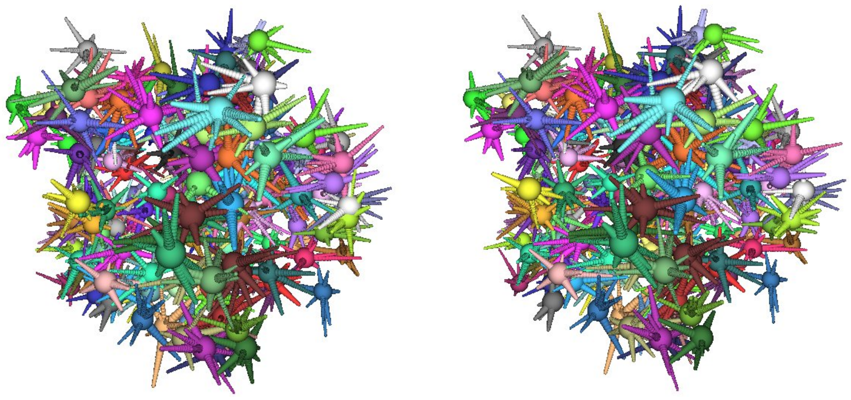
Fig. 7. An example instance with 50 labels of Problem 2: (a) an initial layout, (b) a final layout.

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 size 50, generated 10 instances for each setting, and measured the computation time. See Figure 11 for the average computation time. In this experiment, RigidQN found a layout with an objective function value less than 10^{-9} for all instances. We observed that our algorithm removed almost all overlaps in less



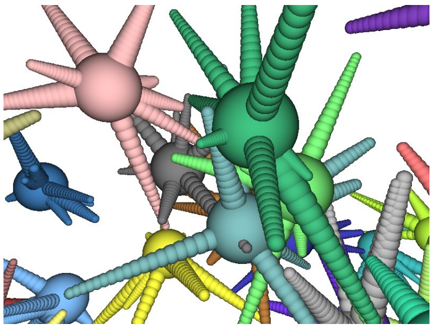
(a) (b)

Fig. 8. An example instance with 100 labels of Problem 2: (a) an initial layout, (b) a final layout.



(a) (b)

Fig. 9. An example instance with 200 labels of Problem 2: (a) an initial layout, (b) a final layout.


(a)

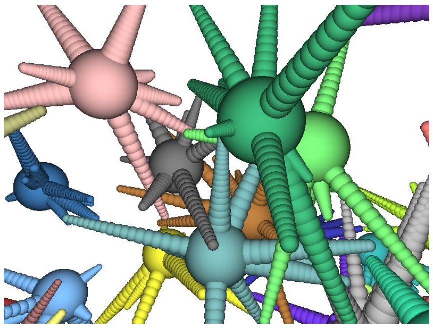

(b)

Fig. 10. Magnified figures of Figure 8: (a) an initial layout, (b) a final layout.

than 10 seconds for the instances with the number of spikes spheres less than or equal to 200.

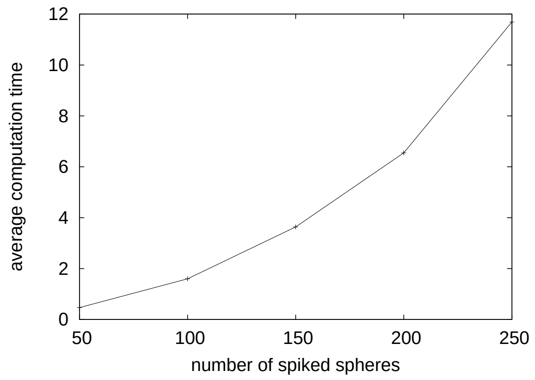


Fig. 11. Average computation time for instances of Problem 2.

5 Conclusion

We presented a new approach for two new variations of label overlap removal problem based on multi-sphere scheme. Our approach is 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.

We applied our algorithm to two new label overlap problems: two dimensional rectangular label with directed-constrained translation (Problem 1) and three dimensional multi-attribute label with translation and rotation (Problem 2). The experimental results showed that our algorithm based on local search algorithm RigidQN was very efficient for label overlap removal. For Problem 1, RigidQN removed almost all overlaps of the labels if the density of labels in the initial layout was not so high, and found a layout in less than one second for the instances with a few thousand circles. For Problem 2, it also removed almost all overlaps of the labels in less than 10 seconds for the instances with less than or equal to 20000 spheres.

Our future work includes more extensive experiments with real world applications. For example, we will consider label overlap removal problem with domain-specific layout constraints including software engineering (such as UML diagram), biology (such as biochemical pathways), and social networks visualization.

References

1. M. Agrawala, Visualizing route maps, Ph.D. thesis, Stanford University, 2002.
2. A. Ahmed, T. Dywer, S. Hong, C. Murray, S. Le and Y. Wu, Visualisation and analysis of large and complex scale-free networks, Proc. of EuroVis 2005, 239-246, 2005.
3. P. K. Agarwal, L. J. Guibas, S. Har-Peled, A. Rabinovitch and M. Sharir, Penetration depth of two convex polytopes in 3D, Nordic Journal of Computing, 7, 2000, 227-240.
4. G. Di Battista, P. Eades, R. Tamassia and I. G. Tollis, Graph Drawing: Algorithms for the Visualization of Graphs, Prentice Hall, 1999.
5. T. Dwyer, K. Marriott and P. J. Stuckey, Fast node overlap removal, Proc. of Graph Drawing 2005, LNCS 3843, 153-164, 2006.
6. E. R. Gansner and S. C. North, Improved force-directed layouts, Proc. of Graph Drawing 1998, LNCS 1547, 364-373, 1999.
7. K. Hayashi, M. Inoue, T. Masuzawa and H. Fujiwara, A layout adjustment problem for disjoint rectangles preserving orthogonal order, Proc. of Graph Drawing 1998, LNCS 1547, 183-197, 1999.
8. X. Huang and W. Lai, Force-transfer: a new approach to removing overlapping nodes in graph layout, Proc. of ACSC 2003, CRPIT 16, 349-358, 2003.
9. T. Imamichi and H. Nagamochi, A multi-sphere scheme for 2D and 3D packing problems, Proc. of Stochastic Local search (SLS) algorithms, LNCS 4638, 207-211, 2007.
10. W. Li, P. Eades and N. Nikolov, Using spring algorithms to remove node overlapping, Proc. of APVIS 2005, CRPIT 45, 131-140, 2005.
11. D. C. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization, Mathematical Programming, 45, 503-528, 1989.
12. K. A. Lyons, H. Meijer and D. Rappaport, Algorithms for cluster busting in anchored graph drawing, Journal of Graph Algorithms and Applications, 2, 1-24, 1998.
13. K. Marriott, P. Stuckey, V. Tam and W. He, Removing node overlapping in graph layout using constrained optimization, Constraints, 8, 143-171, 2003.
14. K. Misue, P. Eades, W. Lai and K. Sugiyama, Layout adjustment and the mental map, Journal of Visual Language and Computing, 6(2), 183-210, 1995.