International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:8, No:5, 2014

An Algorithm for the Map Labeling Problem with Two Kinds of Priorities

Noboru Abe, Yoshinori Amai, Toshinori Nakatake, Sumio Masuda, Kazuaki Yamaguchi

Abstract—We consider the problem of placing labels of the points on a plane. For each point, its position, the size of its label and a priority are given. Moreover, several candidates of its label positions are prespecified, and each of such label positions is assigned a priority. The objective of our problem is to maximize the total sum of priorities of placed labels and their points. By refining a labeling algorithm that can use these priorities, we propose a new heuristic algorithm which is more suitable for treating the assigned priorities.

Keywords—Map labeling, greedy algorithm, heuristic algorithm, priority.

I. Introduction

WE consider the problem of placing labels of the points representing landmarks in a map. Each of these points has a text label that usually indicates its name. For readability purposes, labels can not overlap any other labels or other points. The problem of placing as many labels as possible under this condition is referred to as the map labeling problem. Fig. 1 shows a small example of a solution of this problem, where the black circles and the rectangles represent points and their labels, respectively.

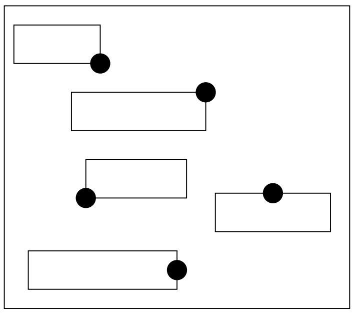

Fig. 1. A small example of label placement

Many algorithms have been proposed to solve this problem [1]. A popular approach is to create several candidate label positions for each point, and then select the final label position from the candidates. We refer to each of these candidate label positions as a label candidate. This approach was adopted by Wagner and Wolff [2]. In their algorithm, label candidates are retained or deleted (if unnecessary) based on four rules L1–L4 and a heuristic procedure. Later, they removed rule L4 from the algorithm [3].

In this paper, we consider the labeling problem in the following situation:

(i) A set of points in a rectangular region R is given. For each point i, its position, the size of its label and a priority $pr_p(i)$ are given.
(ii) For each point, several label candidates are predetermined. Each label candidate l is assigned a priority $pr_c(l)$.

The objective of our problem is to maximize the total sum of priorities of placed labels and their points.

Priority $pr_p(i)$ indicates the importance of a point i, while $pr_c(l)$ indicates the desirability of the position of a label candidate l. The priorities of label candidates can be used to represent the preference about the label positions. For example, it can be used to precede right-hand side label positions. In dynamically changing maps such as online maps, it may be meaningful to keep the label positions around the points in order to maintain the mental map. In such a case, the priorities can be used to precede the current label positions.

Based on the algorithm in [2], Funakawa et al. [4] developed a labeling algorithm with priorities of points. For the case in which label candidates are also assigned priorities, references [3] and [5] presented labeling algorithms. In these papers, the rules in [2] are modified and used. The method in [3] uses rules L1–L3 only, while the algorithms in the other two papers use modified versions of four rules L1– L4.

The algorithm in [5] places the labels of both points and chains, where a chain is a collection of sequential straight line segments representing a feature such as a road or a river. It is not difficult to modify this algorithm so that it works in the abovementioned situation (i) and (ii). Hereafter, we refer to such a modified algorithm as algorithm AKMY. By refining several parts of this algorithm, we propose a new labeling algorithm that is more suitable for treating two kinds of priorities. Furthermore, by comparing it experimentally with the algorithm in [3] and algorithm AKMY, we show the effectiveness of our algorithm.

The remainder of this paper is organized as follows. In Section II, we briefly explain algorithm AKMY. In Section III, we propose a heuristic algorithm for our labeling problem. We show the experimental results in Section IV. Finally, Section V concludes this paper.

Noboru Abe is with the Faculty of Information and Communication Engineering, Osaka Electro-Communication University, Japan (corresponding author to provide e-mail: abe@isc.osakac.ac.jp).
Yoshinori Amai was with the Graduate School of Engineering, Osaka Electro-Communication University, Japan. He is now with Nippon Computer Systems Corp., Japan.
Toshinori Nakatake was with the Faculty of Information Science and Arts, Osaka Electro-Communication University, Japan. He is now with DENSO TECHNO Co., Ltd., Japan.
Sumio Masuda and Kazuaki Yamaguchi are with the Graduate School of Engineering, Kobe University, Japan (e-mail: masuda@kobe-u.ac.jp, ky@kobe-u.ac.jp).

II. Algorithm AKMY

Let $LC(i)$ be the set of all label candidates of point i. For each label candidate l, let $point(l)$ be the point that has l. We define $pr(l)$ to be $pr_p(point(l)) + pr_c(l)$. If a label candidate l of a point overlaps a label candidate l' of another point, we say that l and l' conflict with each other. We denote by X_l the set of all label candidates that conflict with l, and we specify $|X_l|$ as the degree of l.

Algorithm AKMY was developed based on the algorithm in [2]. It uses four rules (designated L1′–L4′) and a heuristic procedure to accept label candidates as label positions and delete unnecessary ones.

Rules L1′–L3′ are as follows. These are the same as the three rules used in the algorithm in [3] that can handle the priorities.

- Rule L1′ : If the degree of a label candidate l of point i is 0, delete all label candidates in $\{l' \mid l' \in LC(i) - \{l\}$ and $pr(l') \leq pr(l)\}$. Then, if $|LC(i)| = 1$, select l as the label position of i.
- Rule L2′ : Let l and l' be label candidates of points i and j, respectively. We call (l, l') a safe pair with priority $pr(l) + pr(l')$, if $X_l \subseteq LC(j) - \{l'\}$ and $X_{l'} \subseteq LC(i) - \{l\}$. If there exists such a pair (l, l'), then delete all label candidates in $LC(i) \cup LC(j)$ that are not contained in any safe pair with priority higher than $pr(l) + pr(l')$.
- Rule L3′ : If a point i has only one label candidate l left and any two label candidates in X_l conflict with each other, delete all label candidates in $\{l' \mid l' \in X_l$ and $pr(l') \leq pr(l)\}$. Then, if the degree of l is zero, select l as the label position of i.

Rule L4′ used in algorithm AKMY is as follows:

- Rule L4′ : If a point i has two label candidates l and l' such that $X_{l'} \supseteq X_l$ and $pr(l') \leq pr(l)$, then delete l'. After the deletion, if i satisfies the condition of L3′, apply L3′ to i.

Algorithm AKMY proceeds in two phases. In phase I, rules L1′–L4′ are iteratively applied to all points in order to reduce the number of label candidates. If the termination condition (defined below) is satisfied at the end of phase I, the algorithm terminates; otherwise, it executes phase II.

Termination Condition: The number of label candidates of each point is at most one and no two label candidates overlap each other.

Phase II executes the following procedure del_cand, which deletes one label candidate based on its priority. This is a modification of the procedure used in [4].

Procedure del_cand

(a) Find a set PT of all label candidates of as-yet unlabeled points that have the maximum number of remaining label candidates.

(b) Let pr_{min} be the minimum value in $\{pr(l) \mid l \in PT\}$.

(c) Find a set $L_{Low} \subseteq PT$ containing all label candidates l such that $pr(l) \leq pr_{min} + th_1$, where th_1 is a non-negative value.

(d) Calculate the $F(l)$ value for each label candidate $l \in L_{Low}$ as follows:

$$F(l) = \sum_{l' \in X_l} \frac{1}{|LC(point(l'))|}.$$

(e) Delete the label candidate with the highest value of $F(\cdot)$ in L_{Low}.

$F(l)$ is a rough estimate of the loss value of the points near l when l is selected as a label position.

After a single execution of del_cand, rules L1′–L4′ are iterated as many times as possible. This process is repeated until the termination condition is satisfied. Then, the remaining label candidates form the solution to the labeling problem with two kinds of priorities.

III. Our algorithm

In this section, we propose a new heuristic algorithm which is suitable for treating priorities, by modifying the rules and procedure del_cand of algorithm AKMY.

A. *Modification of the rules*

First, we modify rules L1′–L4′ of algorithm AKMY. The conditions of rules L1′–L4′ reduce the number of the deletions of label candidates considerably. Hence, we relax the conditions. Let th_2 be a non-negative value. In rules L1′, L3′ and L4′, we replace the expression

$$pr(l') \leq pr(l)$$

with

$$pr(l') \leq pr(l) + th_2.$$

In rule L2′, the label candidates to be deleted are those that are not contained in any safe pair with priority higher than $pr(l) + pr(l') + th_2$. In phase I of our algorithm, the value of th_2 is set to zero, and in phase II, it is set to a positive value C_1.

B. *Modification of procedure del_cand*

Our algorithm adopts the modified version of procedure del_cand.

Assume that $PT = \{l_1, l_2\}$ and $pr(l_1) < pr(l_2) \leq pr(l_1) + th_1$. See Fig.2. Note that $pr_{min} = pr(l_1)$ and $L_{Low} = \{l_1, l_2\}$. Since $F(l_1)$ and $F(l_2)$ have the same value, the original version of the procedure del_cand may first delete the higher-priority candidate l_2. To ensure deletion of the lower-priority candidate, we use the following modified procedure del_cand′.

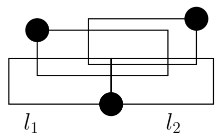

Fig. 2. An example of unsuitable deletion of a label candidate by procedure del_cand

Procedure del_cand′

(a) Find a set PT of all label candidates of as-yet unlabeled points that have the maximum number of remaining label candidates.
(b) Calculate the $F'(l)$ value for each label candidate $l \in PT$ as follows:

$$F'(l) = \sum_{l' \in X_l} \left(pr_c(l') + \frac{pr_p(point(l'))}{|LC(point(l'))|} \right) - \left(pr_c(l) + \frac{pr_p(point(l))}{|LC(point(l))|} \right)$$

(c) Delete the label candidate with the maximum value of $F'(\cdot)$ in PT.

We assume that the worth of each label candidate l is $pr_c(l) + pr_p(point(l))/|LC(point(l))|$. $F'(l)$ roughly estimates the difference between the loss and the gain when l is selected as a label position.

C. Algorithm description

Our algorithm is briefly described below:

[Our algorithm]

```
(1) Delete the label candidates that overlap other points or the boundary of $R$.
(2) Let $ST$ be an empty stack. Push all points onto $ST$.
(3) Set $th_2$ to zero.
(4) While $ST$ is not empty, repeat steps (4a) and (4b):
    (4a) Pop the top element $i$ from $ST$.
    (4b) Apply L1′–L4′ to $i$. Find a set $S$ of the points $j$ such that
         · The rules removed a label candidate of $j$ or decreased the degree of a label candidate of $j$, and
         · $j$ does not exist in $ST$.
         Push the points in $S$ onto $ST$.
(5) If the termination condition is satisfied, then halt.
(6) Set $th_2$ to $C_1$.
(7) Execute procedure del_cand′. Find a set $S'$ of the points $j$ such that
    · del_cand′ removed a label candidate of $j$ or decreased the degree of a label candidate of $j$, and
    · $j$ does not exist in $ST$.
    Push the points in $S'$ onto $ST$. Then, return to step (4).
```

IV. Computational Experiments

We performed computational experiments to compare our algorithm with the algorithm in [3] and algorithm AKMY [5] with respect to the total sum of priorities of placed labels and their points. In Section IV-A below, we explain the experimental conditions used. Next, in Section IV-B, we show our results.

A. Experimental conditions

For use in our experiments, we created many 200-point maps. The points existed only inside a rectangular region R whose size was 1000×1000. Each label was assigned a height of 30 and an individually-determined width in the range 50–100. The priority of each point was individually selected from the range 10–100. For each point, we created eight label candidates as illustrated in Fig.3. The priority of each label candidate was individually selected from the range 1–10.

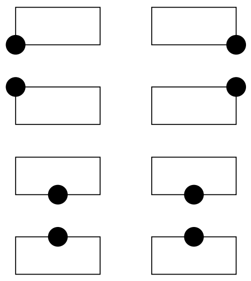

Fig. 3. Eight label candidates for a specified point

In [5], th_1 is set to about 40% of the difference between maximum and minimum value of $pr(\cdot)$. Hence, we set th_1 to 40. C_1 was set to 10, approximately 10% of the difference between maximum and minimum value of $pr(\cdot)$.

We used the C language to implement the algorithms. All computations were performed on a Core 2 Duo E8600 CPU.

B. Experimental results

After step (1) of our algorithm, we computed the following value for each point i assigned at least one label candidate.

$$maxPR(i) = pr_p(i) + \max\{pr_c(l) \mid l \in LC(i)\}.$$

The total sum of $maxPR(i)$ is designated sum_maxPR. After the execution of each algorithm, we computed the following value for each placed label l.

$$labeledPR(point(l)) = pr_p(point(l)) + pr_c(l).$$

We designate the total sum of $labeledPR(point(l))$ as $sum\_labeledPR$. We computed the following value PR_Ratio. We call this value the priority ratio.

$$PR\_Ratio = \frac{sum\_labeledPR}{sum\_maxPR}.$$

We show the experimental results in Table 1. All results are presented as the averages of 100 instances.

TABLE I
Experimental results

algorithm	priority ratio	running time [msec]
algorithm in [3]	86.71%	26.6
algorithm AKMY	88.80%	26.8
our algorithm	90.65%	26.2

With regard to priority ratio, our algorithm is superior to the algorithm AKMY. It also runs slightly faster than the other algorithms. We consider that this improvement was caused by the modified rules L1′–L4′ with th_2, since these rules can delete more label candidates than those without th_2.

V. Conclusion

In this paper, we have considered the labeling problem in which each point and each label candidate had a priority. By refining several parts of the algorithm in [5], we propose a new algorithm that is suitable for treating these priorities. The superior performance of the new algorithm was demonstrated in computational experiments.

References

[1] A. Wolff, The Map-Labeling Bibliography, http://i11www.ira.uka.de/map-labeling/bibliography/.
[2] F. Wagner and A. Wolff, "A combinatorial framework for map labeling problem," Proc. 6th Int'l Symp. Graph Drawing (GD'98), Lecture Notes in Computer Sciences, Volume.1547, pp.316-331, Springer, Berlin, 1998.
[3] F. Wagner, A. Wolff, V. Kapoor and T. Strijk, "Three rules suffice for good label placement," Algorithmica, Volume 30, No.2, pp.334-349, 2001.
[4] K. Funakawa, N. Abe, K. Yamaguchi and S. Masuda, "Algorithms for the map labeling problem with priorities" (in Japanese), IEICE Trans. Fundamentals, Volume J88-A, No.5, pp.677-681, 2005.
[5] N. Abe, M. Kusaki, S. Masuda and K. Yamaguchi, "An algorithm for placing labels of the points and chains on a map," International Journal of Information Science and Computer Mathematics, Volume 4, Issue 2, pp.79-99, 2011.