Discrete Applied Mathematics 239 (2018) 106–118

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Dispersing points on intervals[☆]

Shimin Li [*], Haitao Wang

Department of Computer Science, Utah State University, Logan, UT 84322, USA

ARTICLE INFO

Article history:
Received 24 December 2016
Received in revised form 17 December 2017
Accepted 19 December 2017
Available online 9 January 2018

Keywords:
Dispersing points
Intervals
Lines
Cycles
Algorithms

ABSTRACT

We consider a problem of dispersing points on disjoint intervals on a line. Given n pairwise disjoint intervals sorted on a line, we want to find a point in each interval such that the minimum pairwise distance of these points is maximized. Based on a greedy strategy, we present a linear time algorithm for the problem. Further, we also solve in linear time the cycle version of the problem where the intervals are given on a cycle.

© 2018 Elsevier B.V. All rights reserved.

1. Introduction

The problems of dispersing points have been extensively studied and can be classified to different categories by their different constraints and objectives, e.g., [8,12,15,17,16,21]. In this paper, we consider problems of dispersing points on intervals in linear domains including lines and cycles.

Let $\mathcal{I}$ be a set of n intervals on a line ℓ, and no two intervals of $\mathcal{I}$ intersect. The problem is to find a point in each interval of $\mathcal{I}$ such that the minimum distance of any pair of points is maximized. We assume the intervals of $\mathcal{I}$ are given sorted on ℓ. In this paper we present an $O(n)$ time algorithm for this problem.

As an application of the problem, consider the following scenario. Suppose we are given n pairwise disjoint intervals on ℓ and we want to build a facility on each interval. As the facilities can interfere with each other if they are too close (e.g., if the facilities are hazardous), the goal is to choose locations for these facilities such that the minimum pairwise distance among these facilities is minimized. Clearly, this is an instance of our problem.

We also consider the *cycle version* of the problem where the intervals of $\mathcal{I}$ are given on a cycle $\mathcal{C}$. The intervals of $\mathcal{I}$ are also pairwise disjoint and are given sorted cyclically on $\mathcal{C}$. Note that the distance of two points on $\mathcal{C}$ is the length of the shorter arc of $\mathcal{C}$ between the two points. By making use of our "line version" algorithm, we solve this cycle version problem in linear time as well.

1.1. Related work

To the best of our knowledge, we have not found any previous work on the two problems studied in this paper. Our problems essentially belong to a family of geometric dispersion problems, which are NP-hard in general in two and higher dimensional space. For example, Baur and Fekete [1] studied the problems of distributing a number of points within a

[☆] A preliminary version of this paper appeared in the Proceedings of the 27th International Symposium Algorithms and Computation (ISAAC 2016).
[*] Corresponding author.
E-mail addresses: shiminli@aggiemail.usu.edu (S. Li), haitao.wang@usu.edu (H. Wang).

https://doi.org/10.1016/j.dam.2017.12.028
0166-218X/© 2018 Elsevier B.V. All rights reserved.

polygonal region such that the points are dispersed far away from each other, and they showed that the problems cannot be approximated arbitrarily well in polynomial time, unless P = NP.

Wang and Kuo [21] considered the following two problems. Given a set S of points and a value d, find a largest subset of S in which the distance of any two points is at least d. Given a set S of points and an integer k, find a subset of k points of S to maximize the minimum distance of all pairs of points in the subset. It was shown in [21] that both problems in 2D are NP-hard but can be solved efficiently in 1D. Refer to [2,7,9,10,14] for other geometric dispersion problems. Dispersion problems in various non-geometric settings were also considered [8,12,15–17]. These problems are in general NP-hard; approximation and heuristic algorithms were proposed for them.

On the other hand, problems on intervals usually have applications in other areas. For example, some problems on intervals are related to scheduling because the time period between the release time and the deadline of a job or task in scheduling problems can be considered as an interval on the line. From the interval point of view, Garey et al. [11] studied the following problem on intervals: Given n intervals on a line, determine whether it is possible to find a unit-length sub-interval in each input interval, such that these sub-intervals do not intersect. An $O(n \log n)$ time algorithm was given in [11] for this problem. The optimization version of the above problem was also studied [5,19], where the goal is to find a maximum number of intervals that contain non-intersecting unit-length sub-intervals. Chrobak et al. [5] gave an $O(n^5)$ time algorithm for the problem, and later Vakhania [19] improved the algorithm to $O(n^2 \log n)$ time. The online version of the problem was also considered [4]. Other optimization problems on intervals have also been considered, e.g., see [11,13,18,20].

Our problem may also be related to some problems involving imprecise data. For example, Chambers et al. [3] studied the problem of picking a point from each region (in a set of given geometric regions, e.g., disks) so that the longest edge length of the geometric spanning tree of these points is minimized. Citovsky et al. [6] studied the problem of determining the order of a set of regions so that the TSP tour connecting the points each chosen from a region by an adversary is minimized.

1.2. *Our approach*

For the line version of the problem, our algorithm is based on a greedy strategy. We consider the intervals of $\mathcal{I}$ incrementally from left to right, and for each interval, we will "temporarily" determine a point in the interval. During the algorithm, we maintain a value $d_{\min}$, which is the minimum pairwise distance of the "temporary" points that so far have been computed. Initially, we put a point at the left endpoint of the first interval and set $d_{\min} = \infty$. During the algorithm, the value $d_{\min}$ will be monotonically decreasing. In general, when the next interval is considered, if it is possible to put a point in the interval without decreasing $d_{\min}$, then we put such a point as far left as possible. Otherwise, we put a point on the right endpoint of the interval. In the latter case, we also need to adjust the points that have been determined temporarily in the previous intervals that have been considered. We adjust these points in a greedy way such that $d_{\min}$ decreases the least. A straightforward implementation of this approach can only give an $O(n^2)$ time algorithm. In order to achieve the $O(n)$ time performance, during the algorithm we maintain a "critical list" $\mathscr{L}$ of intervals, which is a subset of intervals that have been considered. This list has some properties that help us implement the algorithm in $O(n)$ time.

We should point out that our algorithm is fairly simple and easy to implement. In contrast, the rationale of the idea is quite involved and it is not an easy task to argue its correctness. Indeed, discovering the critical list is the most challenging work and it is the key idea for solving the problem in linear time.

To solve the cycle version, the main idea is to convert the problem to a problem instance on a line and then apply our line version algorithm. More specifically, we make two copies of the intervals of $\mathcal{I}$ to a line and then apply our line version algorithm on these $2n$ intervals on the line. The line version algorithm will find $2n$ points in these intervals and we show that a particular subset of n consecutive points of them correspond to an optimal solution for the original problem on $\mathcal{C}$.

In the following, we will present our algorithms for the line version in Section 2. The cycle version is discussed in Section 3. Section 4 concludes.

2. The line version

Let $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$ be the set of intervals sorted from left to right on ℓ. For any two points of p and q on ℓ, we use $|pq|$ to denote their distance. Our goal is to find a point p_i in I_i for each $1 \leq i \leq n$, such that the minimum pairwise distance of these points, i.e., $\min_{1 \leq i < j \leq n} |p_i p_j|$, is maximized.

For each interval I_i, $1 \leq i \leq n$, we use l_i and r_i to denote its left and right endpoints, respectively. We assume ℓ is the x-axis. With a little abuse of notation, for any point $p \in \ell$, depending on the context, p may also refer to its coordinate on ℓ. Therefore, for each $1 \leq i \leq n$, it is required that $l_i \leq p_i \leq r_i$.

For simplicity of discussion, we make a general position assumption that no two endpoints of the intervals of $\mathcal{I}$ have the same location (our algorithm can be easily extended to the general case). Note that this implies $l_i < r_i$ for any interval I_i.

The rest of this section is organized as follows. In Section 2.1, we discuss some observations. In Section 2.2, we give an overview of our algorithm. The details of the algorithm are presented in Section 2.3. Finally, we discuss the correctness and analyze the running time in Section 2.4.