Discrete Applied Mathematics 187 (2015) 50–60

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

Computing maximum non-crossing matching in convex bipartite graphs

Danny Z. Chen [a], Xiaomin Liu [a], Haitao Wang [b,*]

[a] Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA
[b] Department of Computer Science, Utah State University, Logan, UT 84322, USA

ARTICLE INFO

Article history:
Received 21 August 2013
Received in revised form 7 February 2015
Accepted 17 February 2015
Available online 17 March 2015

Keywords:
Maximum non-crossing matching
Maximum matching
Convex bipartite graphs
Algorithms
Data structures

ABSTRACT

We consider computing a maximum non-crossing matching in convex bipartite graphs. For a convex bipartite graph of n vertices and m edges, we present an $O(n \log n)$ time algorithm for finding a maximum non-crossing matching in the graph. The previous best algorithm takes $O(m + n \log n)$ time (Malucelli et al., 1993). Since $m = \Theta(n^2)$ in the worst case, our result improves the previous work.

1. Introduction

Developing efficient algorithms for matching problems is an important topic in combinatorics and operations research. In this paper, we study the problem of computing a maximum non-crossing matching in convex bipartite graphs and present an efficient algorithm for it. Roughly speaking, a matching is *non-crossing* if no two edges of the graph in its given embedding intersect each other. The formal problem definition is given below.

1.1. Notation and problem statement

A graph $G = (V, E)$ with vertex set V and edge set E is a *bipartite graph* if V can be partitioned into two subsets A and B (i.e., $V = A \cup B$ and $A \cap B = \emptyset$) such that every edge $e(a, b) \in E$ connects a vertex $a \in A$ and a vertex $b \in B$ (it is often also denoted by $G = (A, B, E)$). A bipartite graph $G = (A, B, E)$ is said to be *convex* on the vertex set B if there is a linear ordering on B, say $B = \{b_1, b_2, \ldots, b_{|B|}\}$, such that for each vertex $a \in A$ and any two vertices b_i and b_j in B with $i < j$, if both b_i and b_j are connected to a by two edges in E, then every vertex $b_t \in B$ with $i \le t \le j$ is connected to a by an edge in E. If G is convex on B, then G is called a *convex bipartite graph*. Fig. 1 shows an example. In this paper, A, B, and E always refer to these sets in a convex bipartite graph $G = (A, B, E)$, and we assume that the vertices in B are ordered as discussed above.

We say that an edge $e(a, b) \in E$ is an *incident edge* of a and b, and a and b are *adjacent* to each other. For each vertex $a_k \in A$, suppose the adjacent vertices of a_k are $b_i, b_{i+1}, \ldots, b_j$ (i.e., all vertices in B from b_i to b_j); then we denote $begin(a_k) = i$ and $end(a_k) = j$.

* Corresponding author.
E-mail addresses: dchen@nd.edu (D.Z. Chen), xliu9@nd.edu (X. Liu), haitao.wang@usu.edu (H. Wang).

http://dx.doi.org/10.1016/j.dam.2015.02.014

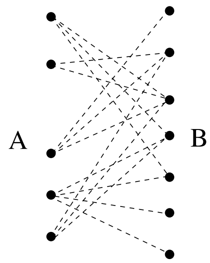

Fig. 1. An example of a convex bipartite graph.

For simplicity, we assume $n = |A| = |B|$. Let $A = \{a_1, a_2, \ldots, a_n\}$. Let $m = |E|$. Note that although m may be $\Theta(n^2)$, the graph G can be represented *implicitly* in $O(n)$ time and $O(n)$ space by giving the two values $begin(a)$ and $end(a)$ for each vertex $a \in A$. A subset $M \subseteq E$ is a *matching* if no two distinct edges in M are connected to the same vertex. Two edges $e(a_i, b_j)$ and $e(a_h, b_l)$ in E are said to be *non-crossing* if either ($i < h$ and $j < l$) or ($i > h$ and $j > l$). Intuitively, suppose we put the two vertex sets A and B on two vertical lines in the plane, respectively, and order them from top to bottom by their indices; if we draw each edge in E as a line segment connecting the corresponding two vertices, then two edges are non-crossing if and only if the two corresponding line segments do not intersect (or one segment is above the other). A matching M is *non-crossing* if no two distinct edges in M intersect. A *maximum non-crossing matching* (MNCM for short) in G is a non-crossing matching M such that no other non-crossing matching in G has more edges than M.

1.2. Related work

Finding maximum matchings in general graphs or bipartite graphs has been well studied [2,3,5,8,10,14]. Glover [7] considered computing maximum matchings in convex bipartite graphs with some industrial applications. Additional matching applications of convex bipartite graphs were given in [12]. A maximum matching in a convex bipartite graph can be obtained in $O(n)$ time [6,12,15]. Liang and Blum [11] gave a linear time algorithm for finding a maximum matching in *circular* convex bipartite graphs. Motivated by applications such as 3-side switch box routing in VLSI design, the problem of finding a maximum *non-crossing* matching (MNCM) in bipartite graphs was studied [9], which can be reduced to computing a longest increasing subsequence in a sequence of size m and thus is solvable in $O(m \log n)$ time [4,18]. An improved $O(m \log \log n)$ time algorithm was given by Malucelli et al. [13] for finding an MNCM in bipartite graphs; further, they showed that in a convex bipartite graph, an MNCM can be found in $O(m + (n - k) \log k)$ time where k is the size of the output MNCM [13], which is $O(m + n \log n)$ time in the worst case. Sweredoski et al. [16] used the MNCM algorithm in [13] for solving genomic sequence problem.

In this paper, we present a new algorithm for computing an MNCM in a convex bipartite graph in $O(n \log n)$ time. Since m can be $\Theta(n^2)$, our result improves the $O(m + n \log n)$ time solution by Malucelli et al. [13]. Our approach is based on the algorithm in [13]; the efficiency of our algorithm hinges on new observations on the problem as well as a data structure for efficiently processing certain frequent operations performed by the algorithm.

The rest of the paper is organized as follows. In Section 2, we briefly discuss the algorithm in [13]. In Section 3, we present our new algorithm. Section 4 concludes the paper.

2. Preliminaries

In this section, we briefly review the algorithm by Malucelli et al. [13], called the *labeling algorithm* (for the full algorithmic and analysis details, see [13]). Our new algorithm given in Section 3 uses some ideas of this labeling algorithm.

For simplicity of discussion, we assume that the vertices of A (resp., B) are ordered on a vertical line in the plane from top to bottom by their indices and each edge in E is represented as a line segment connecting the two corresponding vertices. For any two non-crossing edges $e(a_i, b_j)$ and $e(a_h, b_l)$, we say $e(a_i, b_j)$ is *above* $e(a_h, b_l)$ if $i < h$ and $j < l$, and $e(a_i, b_j)$ is *below* $e(a_h, b_l)$ if $i > h$ and $j > l$.

The labeling algorithm [13] aims to compute a label $L(a, b)$ for each edge $e(a, b) \in E$, which is actually the cardinality of a "partial" MNCM if one considers only the edges of E above and including $e(a, b)$. After the labels for all edges of E are computed, an MNCM can be obtained in additional $O(m)$ time [13]. In order to compute the labels for all edges, the algorithm also computes a label $L(b)$ for each vertex $b \in B$, which is equal to the current maximum label of all incident edges of b whose labels have been computed so far in the algorithm. The value of a vertex label may be increased during the algorithm but is never decreased.

Initially, the label values for all edges of E and all vertices of B are zero. The algorithm considers the vertices in A one by one in their index order. For each vertex $a_i \in A$, there are two procedures for processing it. In the first procedure, for each incident edge $e(a_i, b_j)$ of a_i, the algorithm finds the vertex b_t with the maximum $L(b_t)$ such that $t < j$, and sets $L(a_i, b_j)$ as $L(b_t) + 1$, i.e., $L(a_i, b_j) = 1 + \max\{L(b_t) \mid t < j\}$. After the labels for all incident edges of a_i are computed, the