

Maximum Weighted Matching with Few Edge Crossings for 2-Layered Bipartite Graph

Kazuya Haraguchi* Kotaro Torii Motomu Endo

Abstract

Let c denote a non-negative constant. Suppose that we are given an edge-weighted bipartite graph $G = (V, E)$ with its 2-layered drawing and a family $\mathcal{X} \subseteq E \times E$ of intersecting edge pairs. We consider the problem of finding a maximum weighted matching M^* such that each edge in M^* intersects with at most c other edges in M^* , and that all edge crossings in M^* are contained in \mathcal{X} . In the present paper, we propose polynomial-time algorithms for the cases of $c = 1$ and 2. The time complexities of the algorithms are $O((k + m) \log n + n)$ for $c = 1$ and $O(k^3 + k^2n + m(m + \log n))$ for $c = 2$, respectively, where $n = |V|$, $m = |E|$ and $k = |\mathcal{X}|$.

1 Introduction

Let $G = (A, B, E)$ denote an edge-weighted bipartite graph, where $\{A, B\}$ is the bipartition of the entire vertex set $V = A \cup B$ and $E \subseteq A \times B$ is the edge set. Denoting by $n_A = |A|$ and $n_B = |B|$, we let $A = \{a_1, \dots, a_{n_A}\}$, $B = \{b_1, \dots, b_{n_B}\}$, $n = n_A + n_B$ and $m = |E|$. We abbreviate $(a_i, b_q) \in E$ into $a_i b_q$ for simplicity. The edge weight is given by a function $w : E \rightarrow \mathbb{R}_+$. A subset $M \subseteq E$ of edges is called a *matching* if no two edges in M share an endpoint in common. Denoted by $w(M)$, the weight of a matching M is defined as the sum of edge weights over M , i.e., $w(M) = \sum_{e \in M} w(e)$.

A *2-layered drawing* [2, 18, 19] of a bipartite graph G is a 2D drawing of G such that $a_1, \dots, a_{n_A} \in A$ and $b_1, \dots, b_{n_B} \in B$ are put on two horizontal lines as distinct points from left to right, respectively, and that every edge is drawn as a straight line segment between the endpoints. Two edges $e = a_i b_q$ and $e' = a_j b_p$ make a *crossing* or *intersect* if either $(i < j$ and $p < q)$ or $(i > j$ and $p > q)$ holds.

In our research, we study the problem of computing a maximum weighted (max-weighted for short) matching under a constraint such that only a small number of edge crossings are admitted. The constraint we take up here is that each matching edge may intersect with at most c other matching edges, where c is a non-negative constant.

We formulate the problem in a more general setting. For $e, e' \in E$, we call $\{e, e'\}$ a *crossing pair* if e and e' make a crossing. Let \mathcal{X}_G denote the set of all crossing pairs in G . For input, we accept a subset $\mathcal{X} \subseteq \mathcal{X}_G$ as well as G and w , where \mathcal{X} is the set of crossing pairs that are admitted to make crossings. Let us call \mathcal{X} an *admissible set* and a crossing pair in \mathcal{X} an *admissible pair*. A matching M is called *at-most- c -crossings-per-edge (c -CPE)* if each edge in M makes at most c crossings along with other edges in M and every crossing pair that appears in M belongs to \mathcal{X} .

We formalize the *max-weighted c -CPE matching problem (MW- c -CPEMP)* as follows.

*Corresponding author. E-mail: haraguchi@res.otaru-uc.ac.jp

Max-weighted c -CPE matching problem (MW- c -CPEMP)

Input: A bipartite graph $G = (A, B, E)$ along with its 2-layered drawing, a positive edge weight function $w : E \rightarrow \mathbb{R}_+$, and an admissible set $\mathcal{X} \subseteq \mathcal{X}_G$.

Output: A c -CPE matching $M^* \subseteq E$ that maximizes $w(M^*)$.

For example, when $\mathcal{X} = \mathcal{X}_G$, we are asked to compute a max-weighted c -CPE matching such that any crossing pair may appear. When $c = 0$ or $\mathcal{X} = \emptyset$, the problem asks for a max-weighted non-crossing matching since no crossing pair is admitted.

In the present paper, we propose polynomial time algorithms for the MW- c -CPEMP with $c \in \{1, 2\}$. Our approach reduces the MW- c -CPEMP to what we call the *non-contact trapezoid selection problem (NTSP)*. We then solve the reduced NTSP problem by an algorithm named SELECTTRAPE, which is an extension of the Malucelli et al.'s $O(m \log n)$ -time algorithm for the MW-0-CPEMP [10]. The time complexities of the proposed algorithms are $O((k + m) \log n + n)$ for $c = 1$ and $O(k^3 + k^2n + m(m + \log n))$ for $c = 2$ respectively, where $k = |\mathcal{X}|$.

The paper is organized as follows. We describe our motivation and related work in Section 2. In Section 3, we introduce the NTSP and present the algorithm SELECTTRAPE. We then explain how to reduce the MW- c -CPEMP to the NTSP in Section 4, followed by concluding remarks in Section 5.

2 Background

2.1 Motivation

Plant chronobiologists would like to compare gene expression dynamics at the individual level (i.e., macro level) with the single cell level (i.e., micro level) along the same time axis. However, there is a technically hard issue. Conventional microarray or RNA-sequencing can easily measure individual gene expression patterns in actual time-series but have limited spatial resolutions. On the other hand, single-cell transcriptome techniques have ultimate spatial resolution of gene expression analysis, but most techniques are requiring destruction of cells to perform single cell transcriptome and thus actual time-series analysis is impossible. Thus, to provide an analytic tool to achieve higher spatiotemporal resolution was required.

As an alternative, single cell analysis often uses *pseudo time-series reconstruction* for revealing cell-state transition (e.g., [17, 23]). Pseudo time reconstruction is a process that orders cells transcriptome on a hypothetical time axis, along which they show continuous changes in the transcriptome. However, ordinal scale-based pseudo time-series will not provide any time information so that it is impossible to analyze circadian rhythm, for example, in a single cell resolution.

We have hypothesized that timing of significant gene expression peak on the pseudo time-series is comparable to that on the actual time-series. In our recent work [22], we formulated the problem of estimating the actual time of cell expressions as the MW-0-CPEMP. We considered a 2-layered drawing of a complete bipartite graph $G = (A, B, E)$ such that A is the set of individual expression records that are sorted in the actual-time order, and B is the set of cell expression records that are sorted in a hypothetical order. We weighted each edge $a_i b_q \in E$ by a heuristic method that evaluates how a_i and b_q are likely to match. Solving the MW-0-CPEMP on G , we estimated the

actual time of a gene expression b_q by the time of the individual expression a_i to which b_q is matched. We imposed the non-crossing constraint to preserve the vertex orders. We observed that the model can be a useful tool for actual-time estimation, compared with conventional actual-time estimation methods.

However, the non-crossing constraint is not necessarily a hard constraint since cell expression records may contain noise in practice. In pursuit of alternative models, we study the MW- c -CPEMP for a constant $c \geq 1$.

2.2 Related Work

The MW-0-CPEMP is an extension of the longest common subsequence problem on given two sequences [5]. It has an application in the sequence alignment problem that appears in bioinformatics [24] and in natural language processing [12].

Knauer et al. [9] studied the problem of finding a subgraph that has few edge crossings; given a graph (not necessarily bipartite) and its geometric drawing (i.e., every vertex is specified by a 2D point, all edges are drawn as straight line segments, and no two edges overlap or intersect at a vertex), we are asked to find a subgraph of a certain class that makes the minimum number of edge crossings. They showed that, for spanning trees, s - t paths, cycles, matchings of a fixed size, and 1- or 2-factors, it is NP-hard to approximate the minimum number of edge crossings within a factor of $z^{1-\epsilon}$ for any $\epsilon > 0$, where z denotes the number of edge crossings in the given graph. They also presented fixed-parameter algorithms to decide whether there is a non-crossing subgraph of one of the above graph classes, where z is used as the parameter.

The non-crossing (or crossing-free) constraint has been considered for some problems of finding an “optimal” subgraph. It is Malucelli et al. [10] who first studied the algorithmic aspect of the MW-0-CPEMP explicitly. For the edge-unweighted case, they provided a polynomial-time algorithm that runs in $O(m \log \log n)$ time or in $O(m + \min\{n\mu, m \log \mu\})$ time, where μ denotes the cardinality of a maximum 0-CPE matching. They also extended the algorithm to the edge-weighted case, which yields an $O(m \log n)$ time algorithm. A bipartite graph is *convex* if, for every $a_i \in A$, $a_i b_p, a_i b_q \in E$ ($p \leq q$) implies $a_i b_{p'} \in E$ for all $p \leq p' \leq q$. For the MW-0-CPEMP in edge-unweighted convex bipartite graphs, Chen et al. [4] presented an algorithm whose running time is $O(n \log n)$. Carlsson et al. [3] considered the Euclidean non-crossing bipartite matching problem, where each vertex is represented by a 2D point. The objective is to find a non-crossing perfect matching whose longest edge is minimized. They showed that the problem is NP-hard in general, but that it is polynomially-solvable in some special cases. More recently, Altinel et al. [1] showed that the minimum cost non-crossing flow problem on a layered network is NP-hard. Ruangwises and Itoh [15] studied the stable marriage problem under the non-crossing constraint, showing that there exists a weakly stable non-crossing matching for any instance.

The *conflict pair constraint* (or *negative disjunctive constraint*) is a generalization of the non-crossing constraint. Represented by a *conflict graph* $\hat{G} = (E, \mathcal{C})$, this constraint prohibits a solution from including two edges $e, e' \in E$ such that $\{e, e'\} \in \mathcal{C}$. The minimum cost perfect matching problem with conflict pair constraints is strong NP-hard for a general graph G even if the conflict graph \hat{G} is a collection of single edges [6] and is NP-hard even for G that consists of 4-cycles [13]; it turns out that the problem is NP-hard for a bipartite graph. For this type of constraint, there are also studies on the transportation problem [20, 21], the minimum spanning tree problem [7, 16, 25], and the max-flow problem [14].

3 Non-contact Trapezoid Selection Problem

To solve the MW- c -CPEMP, we reduce the problem to what we call the non-contact trapezoid selection problem (NTSP). In this section, we define the NTSP and propose an efficient algorithm for it. The algorithm is an extension of the Malucelli et al.'s algorithm [10] for the MW-0-CPEMP.

3.1 Problem Description

Suppose two distinct horizontal lines on the 2D plane. Let $A = \{a_1, \dots, a_{n_A}\}$ and $B = \{b_1, \dots, b_{n_B}\}$ denote vertex sets. We put a_1, \dots, a_{n_A} on the upper line from left to right, and b_1, \dots, b_{n_B} on the lower line from left to right. We are given a collection $\mathcal{T} = \{T_1, \dots, T_z\}$ of weighted trapezoids such that each $T_s \in \mathcal{T}$ is given its weight, denoted by $\omega(T_s)$, and its two upper corners are among A , whereas its two lower corners are among B . We denote by $\lambda_A(T_s)$ (resp., $\gamma_A(T_s)$) the index i (resp., j) of the upper-left corner a_i (resp., upper-right corner a_j). Similarly, we denote by $\lambda_B(T_s)$ (resp., $\gamma_B(T_s)$) the index p (resp., q) of the lower-left corner b_p (resp., lower-right corner b_q). We admit T_s to be a triangle or a line segment. Then $\lambda_A(T_s) \leq \gamma_A(T_s)$ and $\lambda_B(T_s) \leq \gamma_B(T_s)$ hold.

Given $(A, B, \mathcal{T}, \omega)$, the NTSP asks for a max-weighted subcollection $\mathcal{S} \subseteq \mathcal{T}$ such that any $T_s, T_t \in \mathcal{S}$ do not contact each other. Specifically, for any $T_s, T_t \in \mathcal{S}$ with $T_s \neq T_t$ and $\lambda_A(T_s) \leq \lambda_A(T_t)$, it should hold that $\gamma_A(T_s) < \lambda_A(T_t)$ and $\gamma_B(T_s) < \lambda_B(T_t)$.

3.2 Partial Order Based Algorithm

We can solve the NTSP by using the notion of partial order. Let us introduce a binary relation \prec on \mathcal{T} ; For $T_s, T_t \in \mathcal{M}$, we write $T_s \prec T_t$ if $\gamma_A(T_s) < \lambda_A(T_t)$ and $\gamma_B(T_s) < \lambda_B(T_t)$. One easily sees that \prec is a (or an irreflexive) partial order on \mathcal{T} , and thus (\mathcal{T}, \prec) is a partially ordered set (poset). We say that T_s and T_t are *comparable* if either $T_s \prec T_t$ or $T_t \prec T_s$ holds. For a subcollection $\mathcal{C} \subseteq \mathcal{T}$, the poset (\mathcal{C}, \prec) (or \mathcal{C}) is called a *chain* if every $T, T' \in \mathcal{C}$ are comparable. Obviously \mathcal{C} is a feasible solution of the NTSP iff it is a chain.

We represent the poset (\mathcal{T}, \prec) by a directed acyclic graph (DAG). We denote the DAG by $D = (\mathcal{T} \cup \{\phi\}, \mathcal{A})$, where ϕ is the dummy node and $\mathcal{A} = \mathcal{A}^\phi \cup \mathcal{A}^\prec$ is the arc set such that

$$\begin{aligned} \mathcal{A}^\phi &= \{(\phi, T') : T' \in \mathcal{T}\}, \\ \mathcal{A}^\prec &= \{(T, T') \in \mathcal{T} \times \mathcal{T} : T \prec T'\}. \end{aligned}$$

For an arc $(T, T') \in \mathcal{A}$, we define the distance to be $\omega(T')$. Any feasible solution of the NTSP is represented by a path from ϕ . Then we can solve the NTSP by solving the longest path problem on D . The time complexity of this algorithm is $O(z^2)$ because we can construct D and solve the longest path problem in $O(z^2)$ time [5].

3.3 An Efficient Algorithm SELECTTRAPE

We propose a faster algorithm whose time complexity is $O(z \log n + n)$. The proposed algorithm is based on the Malucelli et al.'s algorithm [10] for the MW-0-CPEMP. The MW-0-CPEMP is regarded as a special case of the NTSP such that every trapezoid $T \in \mathcal{T}$ is a line segment, that is, $\lambda_A(T) = \gamma_A(T)$ and $\lambda_B(T) = \gamma_B(T)$. We extend the Malucelli et al.'s algorithm based on this observation.

For two integers i, j with $1 \leq i \leq j \leq n_A$, let us denote $[i, j] = \{i, \dots, j\}$ and $[i] = [1, i]$. We define $A[i, j] \triangleq \{a_i, \dots, a_j\}$ and $A[j] \triangleq A[1, j]$. Similarly, for integers p, q with $1 \leq p \leq q \leq n_B$, we

Algorithm 1: An algorithm SELECTTRAPE to compute the optimal value of a given NTSP instance

Input : An instance $(A, B, \mathcal{T}, \omega)$, where $A = \{a_1, \dots, a_{n_A}\}$, $B = \{b_1, \dots, b_{n_B}\}$,
 $\mathcal{T} = \{T_1, \dots, T_z\}$, and $\omega : \mathcal{T} \rightarrow \mathbb{R}_+$

Output: The max-weight of a subcollection $\mathcal{S} \subseteq \mathcal{T}$ such that every $M_s, M_t \in \mathcal{S}$ ($M_s \neq M_t$) do not contact each other

```

1  $\hat{\mu}_\gamma(q) \leftarrow 0$  for all  $q \in [n_B]$ 
2  $\omega^* \leftarrow -\infty$ 
3 for  $i = 1$  to  $n_A$  do
4   for  $T_s \in \mathcal{T}$  such that  $\lambda_A(T_s) = i$  do
5      $p \leftarrow \lambda_B(T_s)$ 
6      $\mu_{\mathcal{T}}(T_s) \leftarrow \omega(T_s) + \max\{\hat{\mu}_\gamma(q) : q \in [p-1]\}$ 
7   for  $T_t \in \mathcal{T}$  such that  $\gamma_A(T_t) = i$  do
8      $q \leftarrow \gamma_B(T_t)$ 
9      $\hat{\mu}_\gamma(q) \leftarrow \max\{\hat{\mu}_\gamma(q), \mu_{\mathcal{T}}(T_t)\}$ 
10    if  $\hat{\mu}_\gamma(q) > \omega^*$  then  $\omega^* \leftarrow \hat{\mu}_\gamma(q)$ 
11 output  $\omega^*$ 

```

define a subset $B[p, q] \triangleq \{b_p, \dots, b_q\}$ and $B[q] \triangleq B[1, q]$. We say that a trapezoid T is *contained in* $A[i, j] \cup B[p, q]$ if all corners are contained in the vertex subset, that is, $\lambda_A(T), \gamma_A(T) \in [i, j]$ and $\lambda_B(T), \gamma_B(T) \in [p, q]$.

For $T_s \in \mathcal{T}$, we denote by $\mu_{\mathcal{T}}(T_s)$ the max-weight of a feasible solution that has T_s as the rightmost trapezoid (i.e., no T in the solution satisfies $T_s \prec T$). For $(i, q) \in [n_A] \times [n_B]$, we denote by $\mu_\gamma(i, q)$ the max-weight of a feasible solution such that the trapezoids are contained in $A[i] \cup B[q]$ and the lower-right corner of the rightmost trapezoid is exactly b_q . For convenience, we let $\mu_\gamma(0, q) = 0$ for all $q \in [n_B]$. The following lemmas are obvious by the definitions.

Lemma 1 *Suppose that $(A, B, \mathcal{T}, \omega)$ is given. For $T_s \in \mathcal{T}$, we have*

$$\mu_{\mathcal{T}}(T_s) = \omega(T_s) + \max\left\{\mu_\gamma(\lambda_A(T_s) - 1, q) : q \in [\lambda_B(T_s) - 1]\right\}. \quad (1)$$

Lemma 2 *Suppose that $(A, B, \mathcal{T}, \omega)$ is given. For $(i, q) \in [n_A] \times [n_B]$, we have*

$$\mu_\gamma(i, q) = \max\left\{\mu_\gamma(i-1, q), \max_{T_s \in \mathcal{T}: (\gamma_A(T_s), \gamma_B(T_s)) = (i, q)} \{\mu_{\mathcal{T}}(T_s)\}\right\}.$$

In Algorithm 1, we show an algorithm SELECTTRAPE that computes the optimal weight of a given NTSP instance. The algorithm repeats the outer for-loop for $i = 1, \dots, n_A$. We do not store $\mu_\gamma(i, q)$ for all $i \in [n_A]$, but only for the current i . It is stored as $\hat{\mu}_\gamma(q)$. When every $T \in \mathcal{T}$ is a line segment (i.e., $\lambda_A(T) = \gamma_A(T)$ and $\lambda_B(T) = \gamma_B(T)$), the algorithm works exactly in the same way as the Malucelli et al.'s algorithm.

Theorem 1 *Given an instance $(A, B, \mathcal{T}, \omega)$ of the NTSP, the algorithm SELECTTRAPE in Algorithm 1 computes the optimal weight in $O(z \log n + n)$ time and in $O(z + n)$ space, where $z = |\mathcal{T}|$.*

PROOF: We show that $\hat{\mu}_\gamma(q) = \mu_\gamma(i-1, q)$ holds at the beginning of the outer for-loop with respect to i (i.e., line 3). When $i = 1$, we have $\hat{\mu}_\gamma(q) = 0 = \mu_\gamma(0, q)$ by line 1. In line 6, we see that $\mu_{\mathcal{T}}(T_s)$

Algorithm 2: An algorithm to construct an optimal solution of a given NTSP instance

Input : An instance $(A, B, \mathcal{T}, \omega)$, where $A = \{a_1, \dots, a_{n_A}\}$, $B = \{b_1, \dots, b_{n_B}\}$,
 $\mathcal{T} = \{T_1, \dots, T_z\}$, and $\omega : \mathcal{T} \rightarrow \mathbb{R}_+$, the optimal weight ω^* of the instance, and a
function $\mu_{\mathcal{T}} : \mathcal{T} \rightarrow \mathbb{R}_+$ that satisfies (1)

Output: An optimal solution

```

1  $\mathcal{S} \leftarrow \emptyset$ ;  $\alpha \leftarrow \omega^*$ ;  $i \leftarrow n_A$ ;  $q \leftarrow n_B$ 
2 while  $\alpha > 0$  do
3   for  $T_t \in \mathcal{T}$  such that  $\gamma_A(T_t) = i$  and  $\gamma_B(T_t) \leq q$  do
4     if  $\mu_{\mathcal{T}}(T_t) = \alpha$  then
5        $\mathcal{S} \leftarrow \mathcal{S} \cup \{T_t\}$ ;  $\alpha \leftarrow \alpha - \omega(T_t)$ ;  $i \leftarrow \lambda_A(T_t)$ ;  $q \leftarrow \lambda_B(T_t) - 1$ 
6       break the for-loop
7    $i \leftarrow i - 1$ 
8 output  $\mathcal{S}$ 

```

is computed correctly for $T_s \in \mathcal{T}$ with $\lambda_A(T_s) = i$, due to $\hat{\mu}_{\gamma}(q) = \mu_{\gamma}(i-1, q) = \mu_{\gamma}(\lambda_A(T_s) - 1, q)$ (by induction) and Lemma 1. The second inner for-loop (i.e., line 7 to 10) computes the maximum among $\hat{\mu}_{\gamma}(q) = \mu_{\gamma}(i-1, q)$ and $\mu_{\mathcal{T}}(T_s)$ for all $T_s \in \mathcal{T}$ with $(\gamma_A(T_s), \gamma_B(T_s)) = (i, q)$, and substitutes the maximum for $\hat{\mu}_{\gamma}(q)$, which is $\mu_{\gamma}(i, q)$ by Lemma 2. Upon completion of the algorithm, we have $\omega^* = \max_{q \in [n_B]} \{\hat{\mu}_{\gamma}(q)\} = \max_{q \in [n_B]} \{\mu_{\gamma}(n_A, q)\}$, which is the optimal value.

We analyze the computational complexity. Each trapezoid is searched as T_s in the first inner for-loop exactly once, and as T_t in the second inner for-loop exactly once. We can access $T_s \in \mathcal{T}$ with $\lambda_A(T_s) = i$ in line 4 (and $T_t \in \mathcal{T}$ with $\gamma_A(T_t) = i$ in line 7) in $O(1)$ time by executing the bucket sort on \mathcal{T} beforehand. The bucket sort runs in $O(z+n)$ time. We use *priority search tree* [11] to store $(q, \hat{\mu}_{\gamma}(q))$ for $q \in [n_B]$, by which we can take the maximum in line 9 in $O(\log n_B)$ time. We see that the algorithm runs in $O(z \log n + n)$ time. We use $O(z+n)$ space to store $\mu_{\mathcal{T}}(T_s)$ for $T_s \in \mathcal{T}$ and $\hat{\mu}_{\gamma}(q)$ for $q \in [n_B]$. \square

We explain how to construct an optimal solution \mathcal{S}^* . For the rightmost trapezoid T_t in \mathcal{S}^* , it holds that $\mu_{\mathcal{T}}(T_t) = \sum_{T \in \mathcal{S}^*} \omega(T) = \omega^*$. Similarly, for $\mathcal{S} = \mathcal{S}^* \setminus \{T_t\}$ and the rightmost trapezoid $T_{t'}$ in \mathcal{S} , since \mathcal{S} is a max-weighted feasible solution among those having $T_{t'}$ as the rightmost trapezoid, it holds that $\mu_{\mathcal{T}}(T_{t'}) = \sum_{T \in \mathcal{S}} \omega(T) = \omega^* - \omega(T_t)$. Note that, since $T_{t'} \prec T_t$, $\gamma_A(T_{t'}) < \lambda_A(T_t)$ and $\gamma_B(T_{t'}) < \lambda_B(T_t)$ should hold. Then we can construct \mathcal{S}^* by Algorithm 2 after running Algorithm 1. The running time is $O(z+n)$.

4 Algorithm for the MW- c -CPEMP

In this section, we reduce the MW- c -CPEMP to the NTSP, which yields polynomial-time algorithms for the cases of $c = 1$ and 2. Throughout this section, we assume that an instance (G, w, \mathcal{X}) of the MW- c -CPEMP is given for a non-negative constant c . We also assume that \mathcal{X} is expressed by a list of crossing pairs.

Before proceeding, let us mention that the problem is solvable in $O^*(2^k)$ -time as follows, where $k = |\mathcal{X}|$ and $O^*(\cdot)$ is introduced to ignore polynomial factors; For each $\mathcal{Y} \subseteq \mathcal{X}$, let $M_{\mathcal{Y}} = \{e \in E : \exists Y \in \mathcal{Y}, e \in Y\}$. In other words, $M_{\mathcal{Y}}$ is the set of edges that appear in \mathcal{Y} . We check whether $M_{\mathcal{Y}}$ is a c -CPE matching, which can be done in polynomial time. If it is the case, we compute a max-weighted non-crossing matching, say $M'_{\mathcal{Y}}$, on the subgraph that is obtained by removing

$M_{\mathcal{Y}}$ and the extreme points from G . The $M'_{\mathcal{Y}}$ can be computed in polynomial time by using the Malucelli et al's algorithm [10]. The max-weighted c -CPE matching $M_{\mathcal{Y}} \cup M'_{\mathcal{Y}}$ over $\mathcal{Y} \subseteq \mathcal{X}$ is an optimal solution.

4.1 Notations

For $V' \subseteq V$, we define $\lambda_A(V')$ to be the smallest index among the vertices in $A \cap V'$. For convenience, we let $\lambda_A(V')$ be zero when $A \cap V' = \emptyset$. That is,

$$\lambda_A(V') \triangleq \begin{cases} \min_{a_i \in A \cap V'} \{i\} & \text{if } A \cap V' \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

Similarly, we define $\gamma_A(V')$ to be the largest index among the vertices in $A \cap V'$, and for convenience, it is set to $n_A + 1$ if $A \cap V' = \emptyset$. That is,

$$\gamma_A(V') \triangleq \begin{cases} \max_{a_j \in A \cap V'} \{j\} & \text{if } A \cap V' \neq \emptyset, \\ n_A + 1 & \text{otherwise.} \end{cases}$$

Observe that, when $A \cap V' \neq \emptyset$, $\lambda_A(V')$ and $\gamma_A(V')$ represent the indices of the leftmost and rightmost vertices in $A \cap V'$, respectively. We define $\lambda_B(V')$ and $\gamma_B(V')$ in the analogous way.

For an edge subset $E' \subseteq E$, we define $V[E']$ to be the set of extreme points of edges in E' . For simplicity, we write $\lambda_A(V[E'])$ by $\lambda_A(E')$. The notations $\gamma_A(E')$, $\lambda_B(E')$ and $\gamma_B(E')$ are analogous. When E' is a singleton, that is, $E' = \{e\}$ for some edge $e \in E$, we write $\lambda_A(\{e\})$ as $\lambda_A(e)$. In this case, it holds that $\lambda_A(e) = \gamma_A(e)$ and $\lambda_B(e) = \gamma_B(e)$. We write an inequality $\lambda_A(e) \leq \lambda_A(e')$ as $e \leq_A e'$. Using this notation, e and e' intersect if $(e <_A e' \text{ and } e' <_B e)$ or $(e' <_A e \text{ and } e <_B e')$.

Recall that \mathcal{X}_G denotes the set of all possible crossing pairs in G . Observe that each $X \in \mathcal{X}_G$ is a matching that consists of two intersecting edges. We may write $X = \{e, e'\}$ in \mathcal{X}_G as an ordered pair $X = (e, e')$ when we assume $e <_A e'$ (and thus $e' <_B e$). For a matching $M \subseteq E$, we define $\mathcal{X}_G[M]$ to be a set of crossing pairs that appear in M , that is,

$$\mathcal{X}_G[M] \triangleq \{\{e, e'\} \in \mathcal{X}_G : e, e' \in M\}.$$

A matching M is *at-most- c -crossings-per-edge* (c -CPE) if $\mathcal{X}_G[M] \subseteq \mathcal{X}$ holds and each $e \in M$ appears in at most c crossing pairs in $\mathcal{X}_G[M]$. Hence, M is 0-CPE iff $\mathcal{X}_G[M] = \emptyset$.

4.2 Overview

Let \mathcal{M} denote the family of all matchings in G . We regard any $M \in \mathcal{M}$ as a trapezoid T_M that has a_i with $i = \lambda_A(M)$ as the upper-left corner, a_j with $j = \gamma_A(M)$ as the upper-right corner, b_p with $p = \lambda_B(M)$ as the lower-left corner, and b_q with $q = \gamma_B(M)$ as the lower-right corner, and that has the weight $\omega(T_M) = \sum_{e \in M} w(e)$. Then (\mathcal{M}, \prec) is a poset, where \prec is the partial order on a trapezoid collection that we introduced in Section 3.2.

Lemma 3 *For a 2-layered bipartite graph G , let $M \in \mathcal{M}$. For any $e, e' \in M$, exactly one of the following holds:*

- (i) $\{e, e'\} \in \mathcal{X}_G$; and
- (ii) e and e' are comparable.

PROOF: When $e = e'$, (ii) holds. Suppose that $e \neq e'$. If e and e' intersect, then we have $\{e, e'\} \in \mathcal{X}_G$. Otherwise, either $e \prec e'$ or $e' \prec e$ should hold as they do not share endpoints in common. \square

We introduce an auxiliary graph, which we denote by $H = (E, \mathcal{X}_G)$. We call an edge in the underlying graph G a *node* when we use it in the context of H . For $M \in \mathcal{M}$, we denote by $H[M] = (M, \mathcal{X}_G[M])$ the subgraph induced by M . Let \mathcal{C}_M denote the family of connected components in $H[M]$.

Lemma 4 *For a 2-layered bipartite graph G , let $M \in \mathcal{M}$. Then (\mathcal{C}_M, \prec) is a chain.*

PROOF: We show that any $X, Y \in \mathcal{C}_M$ ($X \neq Y$) are comparable. The X and Y are node sets of connected components of $H[M]$. For any $e_X \in X$ and $e_Y \in Y$, e_X and e_Y are not adjacent. Then $\{e_X, e_Y\} \notin \mathcal{X}_G$ holds. By Lemma 3, e_X and e_Y are comparable. We assume that $e_X \prec e_Y$ without loss of generality. Suppose that there is $e'_Y \in Y$ such that $e'_Y \prec e_X$. Since $H[Y]$ is connected, there is a path between e_Y and e'_Y . The path should contain an edge e''_Y that intersects with e_X in G . Then $\{e_X, e''_Y\} \in \mathcal{X}_G$, which contradicts that e_X and e''_Y are not adjacent in H . We see that $e_X \prec e_{Y'}$ holds for any $e_X \in X$ and $e_{Y'} \in Y$ and thus $X \prec Y$ holds. \square

Let $\mathcal{M}_c \subseteq \mathcal{M}$ denote the family of all c -CPE matchings. For a c -CPE matching $M \in \mathcal{M}_c$, it holds that $\mathcal{X}_G[M] \subseteq \mathcal{X}$ and the degree of any node in $H[M]$ is at most c . We call M *connected* if $H[M]$ is connected. By Lemma 4, any $M \in \mathcal{M}_c$ is partitioned into connected c -CPE matchings.

If we are given a family \mathcal{T} of all connected c -CPE matchings (which are regarded as trapezoids), then we can find a max-weighted c -CPE matching by solving the corresponding NTSP. The time complexity of this algorithm is $O(\Gamma + |\mathcal{T}| \log n + n)$ by Theorem 1, where Γ denotes the time for constructing \mathcal{T} . Then, if Γ and $|\mathcal{T}|$ are polynomially bounded, the total running time of the algorithm is also polynomially bounded.

Let us consider how to construct \mathcal{T} . For $M \in \mathcal{M}_c$, the degree of any node in $H[M]$ is at most c . Then, when $c = 0$, we have $\mathcal{T} = E$, that is, the collection of isolated nodes in H . Then $|\mathcal{T}| = m$ holds. When $c = 1$, we have $\mathcal{T} = E \cup \mathcal{X}$, and thus $|\mathcal{T}| = m + k$ holds. For $c \in \{0, 1\}$, the following theorems are immediate.

Theorem 2 (Malucelli et al. [10]) *Given an instance (G, w) of the MW-0-CPEMP, we can find a max-weighted 0-CPE matching in $O(m \log n)$ time and in $O(m + n)$ space.*

Theorem 3 *Given an instance (G, w, \mathcal{X}) of the MW-1-CPEMP, we can find a max-weighted 1-CPE matching in $O((k + m) \log n + n)$ time and in $O(k + m + n)$ space.*

For Theorem 2, the complexity would be $O(m \log n + n)$ if we apply Theorem 1 directly to the analysis. However, the second term $O(n)$ can be dropped since we do not need to execute the bucket sort before running Algorithm 1. When $c = 0$, each trapezoid is an edge itself. The bucket sort is not needed on condition that the graph is represented by an adjacency list.

4.3 Trapezoid Collection for $c = 2$

For a connected 2-CPE matching M , the auxiliary graph $H[M]$ is an isolated point, an edge, a cycle or a path, and $\mathcal{X}_G[M] \subseteq \mathcal{X}$ should hold. The next lemma tells that, when it is a cycle, the length (which is $|M|$) is at most four. We call a cycle an ℓ -*cycle* if the length is ℓ .

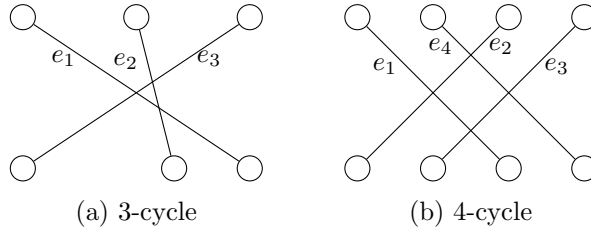


Figure 1: 2-CPE matchings that appear as cycles in the auxiliary graph H

Lemma 5 For a 2-layered bipartite graph G and an admissible set \mathcal{X} , let M be a 2-CPE matching. If $H[M]$ is a cycle, then $|M|$ is at most four.

PROOF: Each edge in M intersects with exactly two other edges in M . Hence $|M| \geq 3$. Let $M = \{e_1, \dots, e_d\}$ ($d \geq 3$). Without loss of generality, we suppose that $e_1 <_A e_t$ holds for all $t \in [2, d]$, that e_2 and e_3 are two edges that intersect with e_1 , and that $e_2 <_A e_3$ holds.

As shown in Figure 1 (a), if e_2 and e_3 intersect, then we see that any of $\{e_1, e_2, e_3\}$ intersects with two others. Since $H[M]$ is a cycle, $M = \{e_1, e_2, e_3\}$ holds. Otherwise (i.e., if e_2 and e_3 do not intersect), as shown in Figure 1 (b), we have $e_1 <_A e_2 <_A e_3$ and $e_2 <_B e_3 <_B e_1$. Since $H[M]$ is a cycle, the edge e_2 intersects with another edge in M , say e_4 . From the definition of e_1 , we have $e_1 <_A e_4$. Since e_1 and e_4 should not intersect, $e_2 <_B e_3 <_B e_1 <_B e_4$ holds. Since e_2 and e_4 intersect, $e_1 <_A e_4 <_A e_2 <_A e_3$ holds. We see that e_3 intersects with e_1 and e_4 , and e_4 intersects with e_2 and e_3 . Any of $\{e_1, \dots, e_4\}$ intersects with two others, and thus we have $M = \{e_1, \dots, e_4\}$. \square

Lemma 6 For a 2-layered bipartite graph G and an admissible set \mathcal{X} , we can enumerate all 3- and 4-cycles in $O(k^2 + m^2)$ time and in $O(m^2)$ space.

PROOF: First, we construct an $m \times m$ intersection matrix \mathcal{I} such that each row/column corresponds to an edge, and that each entry takes 1 (resp., 0) if the corresponding edge pair belongs to \mathcal{X} (resp., does not belong to \mathcal{X}). We can construct \mathcal{I} in $O(m^2)$ time and store it in $O(m^2)$ space.

We can enumerate all 4-cycles in $O(k^2)$ time as follows; for each $X, Y \in \mathcal{X}$, let $X = (e_1, e_3)$ and $Y = (e_4, e_2)$, where we assume $e_1 \leq_A e_4$ without loss of generality. We check whether $X \cup Y$ is a matching such that $\{e_1, e_2\}, \{e_3, e_4\} \in \mathcal{X}$ and $\{e_1, e_4\}, \{e_2, e_3\} \notin \mathcal{X}_G$. If yes, then $X \cup Y$ is a 4-cycle (Figure 1 (b)). The check can be done in $O(1)$ time since whether $\{e, e'\} \in \mathcal{X}$ or not can be identified in $O(1)$ time by using \mathcal{I} . Enumeration of 3-cycles is analogous. \square

There may exist an exponentially large number of paths in H . However, for our purpose, it is sufficient to take into account only $O(k^2)$ paths; Let M be a 2-CPE matching such that $H[M]$ is a path. There are two nodes $e, e' \in M$ whose degrees are one. This means that e and e' appear in exactly one admissible pair in $\mathcal{X}_G[M]$. Suppose that $e <_A e'$ holds without loss of generality. We call $X \in \mathcal{X}_G[M]$ with $e \in X$ (resp., $e' \in X$) the *leftmost* (resp., *rightmost*) *admissible pair* of M . For $X, Y \in \mathcal{X}$, we call a 2-CPE matching M an (X, Y) -*path* if $H[M]$ is a path and X and Y are the leftmost and rightmost admissible pairs of M , respectively. Among all (X, Y) -paths, we have only to take a max-weighted one into account because all (X, Y) -paths form the same trapezoid whose corners are $a_i, a_j, b_p,$ and b_q , where $i = \lambda_A(X)$, $j = \gamma_A(Y)$, $p = \lambda_B(X)$, and $q = \gamma_B(Y)$. We define the size of an (X, Y) -path M to be $|M|$, that is, the number of edges in the matching M . In Figure 2 (a) and (b), we show a 2-CPE matching $M = \{e_1, \dots, e_8\}$ that is an (X, Y) -path

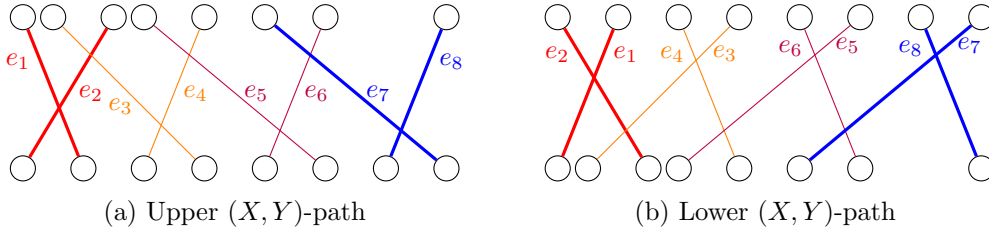


Figure 2: (X, Y) -paths in an underlying graph G ; $X = \{e_1, e_2\}$, $Y = \{e_7, e_8\}$

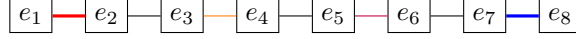


Figure 3: Path $H[M]$ in the auxiliary graph H for (X, Y) -paths M in Figure 2

for $X = \{e_1, e_2\}$ and $Y = \{e_7, e_8\}$. The size of M is eight. We also show the path $H[M]$ in the auxiliary graph H in Figure 3.

For $X, Y \in \mathcal{X}$, we denote by $\rho^*(X, Y)$ the max-weight of an (X, Y) -path. We also denote by $\rho_{\text{odd}}^*(X, Y)$ (resp., $\rho_{\text{even}}^*(X, Y)$) the max-weight of an (X, Y) -path such that the size is odd (resp., even). We let $\rho^*(X, Y)$, $\rho_{\text{odd}}^*(X, Y)$, and $\rho_{\text{even}}^*(X, Y)$ be $-\infty$ when no corresponding path exists. Clearly we have

$$\rho^*(X, Y) = \max\{\rho_{\text{odd}}^*(X, Y), \rho_{\text{even}}^*(X, Y)\}. \quad (2)$$

If $X = Y$, then the path size is two and $\rho^*(X, Y) = w(X)$ holds. If $X \cap Y = \{e\}$, then the path size is three and $\rho^*(X, Y) = w(X) + w(Y) - w(e)$ holds. If $X \cap Y = \emptyset$, then the path size is no less than four.

A max-weighted even-sized (X, Y) -path. We study how to obtain a max-weighted even-sized (X, Y) -path for given $X, Y \in \mathcal{X}$. We design an algorithm that computes $\rho_{\text{even}}^*(X, Y)$ and constructs the path. This strategy is then extended to the odd-size case.

For $X, Y \in \mathcal{X}$, let $X = (e_X, e'_X)$ and $Y = (e_Y, e'_Y)$. We say that an ordered pair (X, Y) is a *link* if one of the followings holds:

- (a) $e_X \prec e_Y$, $e_X \prec e'_Y$, $\{e'_X, e_Y\} \in \mathcal{X}$, and $e'_X \prec e'_Y$.
- (b) $e_X \prec e_Y$, $\{e_X, e'_Y\} \in \mathcal{X}$, $e'_X \prec e_Y$, and $e'_X \prec e'_Y$.

If (X, Y) is a link that satisfies the condition (a) (resp., (b)), then we call it an *upper link* (resp., a *lower link*). In Figure 2 (a), every $(\{e_t, e_{t+1}\}, \{e_{t+2}, e_{t+3}\})$ with $t \in \{1, 3, 5\}$ is an upper link. In Figure 2 (b), every $(\{e_t, e_{t+1}\}, \{e_{t+2}, e_{t+3}\})$ with $t \in \{1, 3, 5\}$ is a lower link.

The following lemma gives a characterization of an even-sized path.

Lemma 7 *For a 2-layered bipartite graph G and an admissible set \mathcal{X} , let $M \in \mathcal{M}_2$ such that $M = \{e_1, \dots, e_{2d}\}$ for an integer $d \geq 2$. If $H[M]$ is a path $e_1 \rightarrow \dots \rightarrow e_{2d}$ such that $e_1 <_A e_{2d}$, then all of $(\{e_{2t-1}, e_{2t}\}, \{e_{2t+1}, e_{2t+2}\})$, $t \in [d-1]$, are either upper links or lower links.*

PROOF: For $t \in [d]$, we denote $X_t = \{e_{2t-1}, e_{2t}\}$. For $s, s' \in [2d]$, if $|s - s'| = 1$, then $\{e_s, e_{s'}\} \in \mathcal{X}_G[M]$ holds, which means that two edges e_s and $e_{s'}$ intersect. Otherwise, since $\{e_s, e_{s'}\}$ does not belong to \mathcal{X}_G , they are comparable (Lemma 3). Among the edges in X_t and X_{t+1} , $t \in [d-1]$, $e_{2t} \in X_t$ and $e_{2t+1} \in X_{t+1}$ intersect, and $\{e_{2t-1}, e_{2t+1}\}$, $\{e_{2t-1}, e_{2t+2}\}$ and $\{e_{2t}, e_{2t+2}\}$ are comparable

pairs. We claim that $e_{2t-1} \prec e_{2t+1}$ should hold; if not so, let $t' \in [d-1]$ denote the smallest index such that $e_{2t'+1} \prec e_{2t'-1}$. This means $e_1 \prec \dots \prec e_{2t'-1}$. We have $e_1 \prec e_{2d}$ by assumption. If $e_{2d} \prec e_{2t'-1}$, there is an edge e_s ($s \in [2, 2t' - 2]$) that intersects with e_{2d} , which contradicts that e_{2d} intersects with only e_{2d-1} . If $e_{2t'-1} \prec e_{2d}$, since $e_{2t'+1} \prec e_{2t'-1}$, there is an edge e_s ($s \in [2t' + 2, 2d - 1]$) that intersects with $e_{2t'-1}$, which contradicts that $e_{2t'-1}$ intersects with only $e_{2t'-2}$ and $e_{2t'}$. We can show that $e_{2t-1} \prec e_{2t+2}$ and $e_{2t} \prec e_{2t+2}$ also hold in the same way.

Now we have two cases: $e_{2t-1} <_A e_{2t}$ or $e_{2t} <_A e_{2t-1}$. In the former case, (X_t, X_{t+1}) is an upper link since the condition (a) holds by $e_X = e_{2t-1}$, $e'_X = e_{2t}$, $e_Y = e_{2t+1}$ and $e'_Y = e_{2t+2}$. Then $e_{2t+1} <_A e_{2t+2}$ also holds. For $t \leq d-2$, (X_{t+1}, X_{t+2}) is also an upper link since the condition (a) holds by $e_X = e_{2t+1}$, $e'_X = e_{2t+2}$, $e_Y = e_{2t+3}$ and $e'_Y = e_{2t+4}$. We see that, if (X_1, X_2) is an upper link, then (X_{t+1}, X_{t+2}) is also an upper link for $t \in [d-2]$. The latter case (i.e., $e_{2t} <_A e_{2t-1}$) is analogous. \square

For $X, Y \in \mathcal{X}$, let M denote an even-sized (X, Y) -path such that the path $H[M]$ is given by $e_1 \rightarrow \dots \rightarrow e_{2d}$ and $e_1 <_A e_{2d}$. We call M an *upper* (resp., a *lower*) (X, Y) -path if all of $(\{e_{2t-1}, e_{2t}\}, \{e_{2t+1}, e_{2t+2}\})$, $t \in [d-1]$, are upper (resp., lower) links. Figure 2 (a) and (b) illustrate these two types of paths. For convenience, we regard that an (X, X) -path is an upper path as well as a lower path. We denote by $\rho_{\text{even}}^{*\uparrow}(X, Y)$ (resp., $\rho_{\text{even}}^{*\downarrow}(X, Y)$) the max-weight of an upper (resp., a lower) (X, Y) -path such that the size is even. We define $\rho_{\text{even}}^{*\uparrow}(X, Y) \triangleq -\infty$ (resp., $\rho_{\text{even}}^{*\downarrow}(X, Y) \triangleq -\infty$) if no such (X, Y) -path exists. Clearly we have

$$\rho_{\text{even}}^*(X, Y) = \max\{\rho_{\text{even}}^{*\uparrow}(X, Y), \rho_{\text{even}}^{*\downarrow}(X, Y)\}. \quad (3)$$

Due to symmetry, we focus on even-sized upper paths. For $X \in \mathcal{X}$, we define $\text{NEXT}(X) \triangleq \{Z \in \mathcal{X} : (X, Z) \text{ is an upper link}\}$. For $Y \in \mathcal{X}$, we define $\text{PREV}(Y) \triangleq \{Z \in \mathcal{X} : Y \in \text{NEXT}(Z)\}$. Moreover, for $j \in [\lambda_A(Y) + 1, \gamma_A(Y) - 1]$, we define the subset $\text{PREV}(Y; j) = \{Z \in \text{PREV}(Y) : \gamma_A(Z) = j\}$. For $X, Y \in \mathcal{X}$ with $X \neq Y$, an even-sized upper (X, Y) -path should contain $Z \in \text{PREV}(Y; j)$ for some $j \in [\lambda_A(Y) + 1, \gamma_A(Y) - 1]$. In other words, (Z, Y) is the ‘‘last’’ upper link on the (X, Y) -path. Then we define $\rho_{\text{even}}^{*\uparrow}(X, Y; j)$ to be the max-weight of an even-sized upper (X, Y) -path such that last upper link on the path, say (Z, Y) , satisfies $\gamma_A(Z) \in [\lambda_A(Y) + 1, j]$. Again, if no such path exists, we define $\rho_{\text{even}}^{*\uparrow}(X, Y; j) \triangleq -\infty$. Obviously we have

$$\rho_{\text{even}}^{*\uparrow}(X, Y; \lambda_A(Y) + 1) \leq \dots \leq \rho_{\text{even}}^{*\uparrow}(X, Y; \gamma_A(Y) - 1) = \rho_{\text{even}}^{*\uparrow}(X, Y).$$

Let $L = (\mathcal{X}, \mathcal{L})$ denote a digraph such that the admissible set \mathcal{X} is the node set and that $\mathcal{L} = \{(Z, Z') \in \mathcal{X} \times \mathcal{X} : Z' \in \text{NEXT}(Z)\}$ is the arc set. Since $\lambda_A(Z) < \lambda_A(Z')$ holds for any $(Z, Z') \in \mathcal{L}$, no cycle exists in L , that is, L is a DAG. We define the subset $\mathcal{X}_X \subseteq \mathcal{X}$ to be $\mathcal{X}_X = \{Y \in \mathcal{X} : \text{there is a path from } X \text{ to } Y \text{ in } L\}$. Clearly, if there is an even-sized upper (X, Y) -path, then $Y \in \mathcal{X}_X$ holds, while the converse does not necessarily hold. For $Y \in \mathcal{X} \setminus \mathcal{X}_X$, we have

$$\rho_{\text{even}}^{*\uparrow}(X, Y; \lambda_A(Y) + 1) = \dots = \rho_{\text{even}}^{*\uparrow}(X, Y; \gamma_A(Y) - 1) = \rho_{\text{even}}^{*\uparrow}(X, Y) = -\infty.$$

For $Y \in \text{NEXT}(X) \subseteq \mathcal{X}_X$, the only even-sized upper (X, Y) -path is $X \cup Y$. Then it holds that;

$$\rho_{\text{even}}^{*\uparrow}(X, Y; j) = \begin{cases} -\infty & \text{if } j \in [\lambda_A(Y) + 1, \gamma_A(X) - 1], \\ w(X) + w(Y) & \text{if } j \in [\gamma_A(X), \gamma_A(Y) - 1]. \end{cases} \quad (4)$$

The following lemma gives a characterization of $\rho_{\text{even}}^{*\uparrow}(X, Y; j)$ for $Y \in \mathcal{X}_X \setminus \text{NEXT}(X)$.

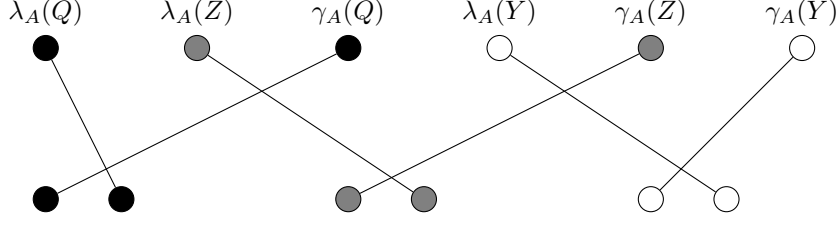


Figure 4: Three admissible pairs Q (black), Z (gray), and Y (white) such that (Q, Z) and (Z, Y) are upper links

Lemma 8 For a 2-layered edge-weighted bipartite graph G and an admissible set \mathcal{X} , let $X, Y \in \mathcal{X}$. If $Y \in \mathcal{X}_X \setminus (\{X\} \cup \text{NEXT}(X))$, then for $j = \lambda_A(Y) + 1$,

$$\rho_{\text{even}}^{*\uparrow}(X, Y; j) = \begin{cases} w(Y) + \max_{Z \in \text{PREV}(Y; j)} \left\{ \rho_{\text{even}}^{*\uparrow}(X, Z; \lambda_A(Y) - 1) \right\} & \text{if } \text{PREV}(Y; j) \neq \emptyset, \\ -\infty & \text{otherwise,} \end{cases} \quad (5)$$

and for any $j \in [\lambda_A(Y) + 2, \gamma_A(Y) - 1]$,

$$\rho_{\text{even}}^{*\uparrow}(X, Y; j) = \begin{cases} \max \left\{ \rho_{\text{even}}^{*\uparrow}(X, Y; j - 1), w(Y) + \max_{Z \in \text{PREV}(Y; j)} \left\{ \rho_{\text{even}}^{*\uparrow}(X, Z; \lambda_A(Y) - 1) \right\} \right\} & \text{if } \text{PREV}(Y; j) \neq \emptyset, \\ \rho_{\text{even}}^{*\uparrow}(X, Y; j - 1) & \text{otherwise.} \end{cases} \quad (6)$$

PROOF: Suppose that $j = \lambda_A(Y) + 1$. If $\text{PREV}(Y; j) = \emptyset$, then there is no even-sized upper (X, Y) -path such that the last link (Z, Y) satisfies $\gamma_A(Z) = j$. Hence we have $\rho_{\text{even}}^{*\uparrow}(X, Y; j) = -\infty$.

We consider the case of $\text{PREV}(Y; j) \neq \emptyset$. Suppose that an even-sized upper (X, Y) -path exists. There is $Z \in \text{PREV}(Y; j)$ such that (Z, Y) is the last link. Let M_Z^* be a max-weighted (X, Y) -path among those having (Z, Y) as the last link. We partition M_Z^* into $M_Z^* = M_Z \cup Y$, where M_Z is a max-weighted even-sized upper (X, Z) -path. Since $Z \neq X$ by $Y \in \mathcal{X}_X \setminus (\{X\} \cup \text{NEXT}(X))$, there is $Q \in \mathcal{X}$ such that (Q, Z) is the last upper link of M_Z (Figure 4). No edge in Q should intersect with any edge in Y , and it holds that $\gamma_A(Q) \in [\lambda_A(Z) + 1, \lambda_A(Y) - 1]$. Hence we have

$$\rho_{\text{even}}^{*\uparrow}(X, Y; j) = \max_{Z \in \text{PREV}(Y; j)} \{w(M_Z^*)\} = w(Y) + \max_{Z \in \text{PREV}(Y; j)} \{w(M_Z)\}$$

and $w(M_Z) = \rho_{\text{even}}^{*\uparrow}(X, Z; \lambda_A(Y) - 1)$. We see that Eq. (5) determines $\rho_{\text{even}}^{*\uparrow}(X, Y; j)$ correctly. If no even-sized upper (X, Y) -path exists, then $\rho_{\text{even}}^{*\uparrow}(X, Z; \lambda_A(Y) - 1) = -\infty$ holds for all $Z \in \text{PREV}(Y; j)$ by induction. We have $\rho_{\text{even}}^{*\uparrow}(X, Y; j) = -\infty$ by (5).

The proof for $j \in [\lambda_A(Y) + 2, \gamma_A(Y) - 1]$ is analogous. \square

By Lemma 8, for a given $X \in \mathcal{X}$, we can compute $\rho_{\text{even}}^{*\uparrow}(X, Y)$ for all $Y \in \mathcal{X}_X$ with $Y \neq X$ as follows. First, we compute the value $\rho_{\text{even}}^{*\uparrow}(X, Y; j)$ for all $Y \in \text{NEXT}(X)$ and $j \in [\lambda_A(Y) + 1, \gamma_A(Y) - 1]$ by (4). Note that $\rho_{\text{even}}^{*\uparrow}(X, Y) = \rho_{\text{even}}^{*\uparrow}(X, Y; \gamma_A(Y) - 1)$ holds. Then, if there is $Y \in \mathcal{X}_X$ such that $\rho_{\text{even}}^{*\uparrow}(X, Y)$ has not been determined and $\rho_{\text{even}}^{*\uparrow}(X, Z)$ has been determined for all $Z \in \text{PREV}(Y)$, we compute the value of $\rho_{\text{even}}^{*\uparrow}(X, Y; j)$ for all $j \in [\lambda_A(Y) + 1, \gamma_A(Y) - 1]$ by (5) and (6).

We summarize the algorithm as `EVENUPPER` in Algorithm 3. The binary flag $\delta(Y)$ is introduced to represent whether $\rho_{\text{even}}^{*\uparrow}(X, Y)$ has been determined or not.

Algorithm 3: An algorithm EVENUPPER to compute $\rho_{\text{even}}^{*\uparrow}(X, Y)$ for a given $X \in \mathcal{X}$ and all $Y \in \mathcal{X}_X \setminus \{X\}$

Input : A 2-layered bipartite graph G , a non-empty admissible set \mathcal{X} , and $X \in \mathcal{X}$

Output: The max-weight $\rho_{\text{even}}^{*\uparrow}(X, Y)$ of an even-sized upper (X, Y) -path for all $Y \in \mathcal{X}_X \setminus \{X\}$

```

1 for  $Y \in \mathcal{X}_X \setminus \{X\}$  do
2   if  $Y \in \text{NEXT}(X)$  then
3     Compute  $\rho_{\text{even}}^{*\uparrow}(X, Y; j)$  by (4) for all  $j \in [\lambda_A(Y) + 1, \gamma_A(Y) - 1]$ 
4      $\delta(Y) \leftarrow \text{TRUE}$ 
5   else  $\delta(Y) \leftarrow \text{FALSE}$ 
6 while there is  $Y \in \mathcal{X}_X \setminus \{X\}$  such that  $\delta(Y) = \text{FALSE}$  and  $\delta(Z) = \text{TRUE}$  for all
    $Z \in \text{PREV}(Y)$  do
7   Compute  $\rho_{\text{even}}^{*\uparrow}(X, Y; j)$  by (5) and (6) for all  $j \in [\lambda_A(Y) + 1, \gamma_A(Y) - 1]$ 
8    $\delta(Y) \leftarrow \text{TRUE}$ 
9 output  $\rho_{\text{even}}^{*\uparrow}(X, Y; \gamma_A(Y) - 1)$  as  $\rho_{\text{even}}^{*\uparrow}(X, Y)$  for all  $Y \in \mathcal{X}_X \setminus \{X\}$ 

```

Lemma 9 For a 2-layered edge-weighted bipartite graph G , a non-empty admissible set \mathcal{X} and an admissible pair $X \in \mathcal{X}$, the algorithm EVENUPPER in Algorithm 3 computes $\rho_{\text{even}}^{*\uparrow}(X, Y)$ for all $Y \in \mathcal{X}_X \setminus \{X\}$ in $O(k^2 + kn)$ time and space.

PROOF: For $Y \in \text{NEXT}(X)$, $\rho_{\text{even}}^{*\uparrow}(X, Y; j)$ is computed in line 3. We see that the algorithm visits all $Y \in \mathcal{X}_X \setminus (\{X\} \cup \text{NEXT}(X))$ by induction with respect to the length of the longest path from X . When Y is visited, $\rho_{\text{even}}^{*\uparrow}(X, Z; \lambda_A(Y) - 1)$ is already computed for all $Z \in \text{PREV}(Y)$. Then $\rho_{\text{even}}^{*\uparrow}(X, Y; j)$ is computed correctly in line 7 for all $j \in [\lambda_A(Y) + 1, \gamma_A(Y) - 1]$ (Lemma 8).

The algorithm EVENUPPER can be implemented as follows. For preprocessing, we construct the DAG L and the family \mathcal{X}_X , which takes $O(k^2)$ time. During the execution, we maintain all Y that satisfy the condition of line 6 in a queue. Every time $\delta(Y)$ is set to TRUE (i.e., lines 4 and 8), we check whether each $Y' \in \text{NEXT}(Y) \setminus \text{NEXT}(X)$ satisfies the condition of line 6; if yes, we insert Y' to the queue. We can do the check over all $Y \in \mathcal{X}_X \setminus \{X\}$ in $O(k^2)$ amortized time since $\sum_Y |\text{NEXT}(Y)| = O(k^2)$. For $Y \in \mathcal{X}_X \setminus (\{X\} \cup \text{NEXT}(X))$ and $j \in [\lambda_A(Y) + 1, \gamma_A(Y) - 1]$, we can compute $\rho_{\text{even}}^{*\uparrow}(X, Y; j)$ in $O(|\text{PREV}(X, Y; j)|)$ time. We obtain $\rho_{\text{even}}^{*\uparrow}(X, Y)$ by computing $\rho_{\text{even}}^{*\uparrow}(X, Y; j)$ for all j , which takes $\sum_j O(|\text{PREV}(X, Y; j)|) = O(|\text{PREV}(X, Y)|)$ time. Therefore, we can compute $\rho_{\text{even}}^{*\uparrow}(X, Y)$ for all Y in $O(k^2 + kn)$ time since every Y is inserted to the queue exactly once, $\sum_Y O(|\text{PREV}(X, Y)|) = O(k^2)$, and there are $O(kn)$ entries for $\rho_{\text{even}}^{*\uparrow}(X, Y; j)$. For the space complexity, we use $O(k^2)$ space for L , and $O(kn)$ space for all $\rho_{\text{even}}^{*\uparrow}(X, Y; j)$ and $\delta(Y)$. \square

The algorithm can be used to construct a max-weighted even-sized upper (X, Y) -path.

Lemma 10 For a 2-layered edge-weighted bipartite graph G , a non-empty admissible set \mathcal{X} and an admissible pair $X \in \mathcal{X}$, we can construct a max-weighted even-sized upper (X, Y) -path for all $Y \in \mathcal{X}$ in $O(k^2 + kn)$ time and space.

PROOF: For $Y \in \mathcal{X}_X \setminus (\{X\} \cup \text{NEXT}(X))$ and $j \in [\lambda_A(Y) + 1, \gamma_A(Y) - 1]$, let us store the maximizer $Z = Z^*$ in (5) and (6) as $\chi(Y; j)$. Specifically, for $j = \lambda_A(Y) + 1$, we let $\chi(Y; j) \leftarrow Z^*$

if $\text{PREV}(Y; j) \neq \emptyset$, and otherwise, we let $\chi(Y; j) \leftarrow \text{NULL}$. For other j , we let $\chi(Y; j) \leftarrow Z^*$ if $\text{PREV}(Y; j) \neq \emptyset$ and $\rho_{\text{even}}^{*\uparrow}(X, Y; j-1) < w(Y) + \rho_{\text{even}}^{*\uparrow}(X, Z^*; \lambda_A(Y) - 1)$, and otherwise, we let $\chi(Y; j) \leftarrow \chi(Y; j-1)$. Observe that, if $Z = \chi(Y; \gamma_A(Y) - 1)$ is not NULL , then (Z, Y) is the last link of a max-weighted even-sized upper (X, Y) -path, and otherwise, no such path exists.

Upon completion of the algorithm, if $Y = X$, the only (X, Y) -path is X itself. If $Y \in \text{NEXT}(X)$, then $X \cup Y$ is the required path. If $Y \in \mathcal{X}_X \setminus (\{X\} \cup \text{NEXT}(X))$ and $\rho_{\text{even}}^{*\uparrow}(X, Y) \neq -\infty$, then we can construct a max-weighted even-sized upper (X, Y) -path by tracing $\chi(Y, \gamma_A(Y) - 1)$, which requires $O(n)$ time since the path size is at most n . For all the other Y , there is no even-sized upper (X, Y) -path. The algorithm runs in $O(k^2 + kn)$ time and space (Lemma 9). The construction of paths can be done in $O(kn)$ time. We can store $\chi(Y; j)$ in $O(kn)$ space. \square

We can derive the similar results for even-sized lower paths due to symmetry.

Lemma 11 *For a 2-layered edge-weighted bipartite graph G , a non-empty admissible set \mathcal{X} and an admissible pair $X \in \mathcal{X}$, we can construct a max-weighted even-sized lower (X, Y) -path for all $Y \in \mathcal{X}$ in $O(k^2 + kn)$ time and space.*

Lemma 12 *For a 2-layered edge-weighted bipartite graph G , a non-empty admissible set \mathcal{X} , we can compute a max-weighted even-sized (X, Y) -path and its weight $\rho_{\text{even}}^*(X, Y)$ for all $X, Y \in \mathcal{X}$ in $O(k^3 + k^2n)$ time and in $O(k^2 + kn)$ space.*

PROOF: By Lemmas 9, 10 and 11, given $X \in \mathcal{X}$, we can compute max-weighted even-sized upper and lower (X, Y) -paths and their weights (i.e., $\rho_{\text{even}}^{*\uparrow}(X, Y)$ and $\rho_{\text{even}}^{*\downarrow}(X, Y)$) for all $Y \in \mathcal{X}$ in $O(k^2 + kn)$ time and space. The path having a larger weight is the required path by (3). Since $|\mathcal{X}| = k$, we have the time complexity $O(k(k^2 + kn)) = O(k^3 + k^2n)$. \square

A max-weighted odd-sized (X, Y) -path. We compute a max-weighted odd-sized (X, Y) -path for given $X, Y \in \mathcal{X}$ by extending the strategy for the even case that we explained so far.

We consider how to compute the max-weight $\rho_{\text{odd}}^*(X, Y)$. Observe that the minimum odd size of an (X, Y) -path is three. Let $X = (e_X, e'_X) \in \mathcal{X}$ and $Z = (e_Z, e'_Z) \in \mathcal{X}$. We say that an ordered pair (X, Z) is a *wedge* if one of the followings holds:

- (a) $e_X = e_Z$ and $e'_X \prec e'_Z$.
- (b) $e_X \prec e_Z$ and $e'_X = e'_Z$.

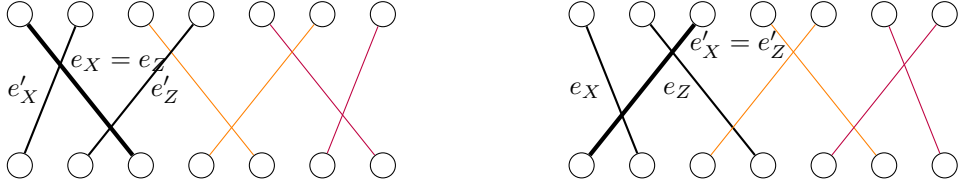
If (X, Z) is a wedge that satisfies (a) (resp., (b)), then we call it an *upper wedge* (resp., a *lower wedge*). See Figure 5. Analogously to Lemma 7, we can show that any larger odd-sized path is obtained by connecting upper links to an upper wedge or lower links to a lower wedge.

Then for $X, Y \in \mathcal{X}$, we can compute $\rho_{\text{odd}}^*(X, Y)$ in a similar fashion to the even case. We denote by $\rho_{\text{odd}}^{*\uparrow}(X, Y)$ (resp., $\rho_{\text{odd}}^{*\downarrow}(X, Y)$) the max-weight of an odd-sized upper (resp., a lower) (X, Y) -path. We define $\rho_{\text{odd}}^{*\uparrow}(X, Y) \triangleq -\infty$ (resp., $\rho_{\text{odd}}^{*\downarrow}(X, Y) \triangleq -\infty$) if no such path exists. Clearly we have

$$\rho_{\text{odd}}^*(X, Y) = \max\{\rho_{\text{odd}}^{*\uparrow}(X, Y), \rho_{\text{odd}}^{*\downarrow}(X, Y)\}.$$

Let us focus on upper paths. The analysis is different from the even case in that wedges should be taken into account in the current case. For $X, Z \in \mathcal{X}$, we define $\text{NEXT}_X(Z)$ as follows;

$$\text{NEXT}_X(Z) \triangleq \begin{cases} \{Y \in \mathcal{X} : (Z, Y) \text{ is an upper wedge}\} & \text{if } Z = X, \\ \{Y \in \mathcal{X} : (Z, Y) \text{ is an upper link}\} & \text{otherwise.} \end{cases}$$



(a) An upper wedge followed by upper links (b) A lower wedge followed by lower links

Figure 5: Wedges (X, Y) and odd-sized paths; a thick edge indicates the edge that is contained in both X and Y

For $Y \in \mathcal{X}$, we define $\text{PREV}_X(Y) \triangleq \{Z \in \mathcal{X} : Y \in \text{NEXT}_X(Z)\}$ and for $j \in [\lambda_A(Y) + 1, \gamma_A(Y) - 1]$, $\text{PREV}_X(Y; j) \triangleq \{Z \in \text{PREV}_X(Y) : \gamma_A(Z) = j\}$. We define $\rho_{\text{odd}}^{*\uparrow}(X, Y; j)$ to be the max-weight of an odd-sized upper (X, Y) -path such that $\gamma_A(Z) \in [\lambda_A(Y) + 1, j]$ holds, where Z is an admissible pair on the path that satisfies $Y \in \text{NEXT}_X(Z)$. If no such path exists, we define $\rho_{\text{odd}}^{*\uparrow}(X, Y; j) \triangleq -\infty$.

We consider a digraph $L_X = (\mathcal{X}, \mathcal{L}_X)$ such that the node set is \mathcal{X} and the arc set \mathcal{L}_X is defined by $\mathcal{L}_X = \{(Z, Z') \in \mathcal{X} \times \mathcal{X} : Z' \in \text{NEXT}_X(Z)\}$. We see that L_X is a DAG. Let $\mathcal{Y}_X = \{Y \in \mathcal{X} : \text{there is a path from } X \text{ to } Y \text{ in } L_X\}$. Whenever an odd-sized upper (X, Y) -path exists, $Y \in \mathcal{Y}_X$ holds. Then we have $\rho_{\text{odd}}^{*\uparrow}(X, Y; j) = -\infty$ for $Y \in \mathcal{X} \setminus \mathcal{Y}_X$ and $j \in [\lambda_A(Y) + 1, \gamma_A(Y) - 1]$, and for $Y \in \text{NEXT}_X(X) \subseteq \mathcal{Y}_X$,

$$\rho_{\text{odd}}^{*\uparrow}(X, Y; j) = \begin{cases} -\infty & \text{if } j \in [\lambda_A(Y) + 1, \gamma_A(X) - 1], \\ w(X \cup Y) & \text{if } j \in [\gamma_A(X), \gamma_A(Y) - 1]. \end{cases}$$

For $Y \in \mathcal{Y}_X \setminus (\{X\} \cup \text{NEXT}_X(X))$, we have the following lemma, which is analogous to Lemma 8 in the even case.

Lemma 13 *For a 2-layered edge-weighted bipartite graph G and an admissible set \mathcal{X} , let $X, Y \in \mathcal{X}$. If $Y \in \mathcal{Y}_X \setminus (\{X\} \cup \text{NEXT}_X(X))$, then for $j = \lambda_A(Y) + 1$,*

$$\rho_{\text{odd}}^{*\uparrow}(X, Y; j) = \begin{cases} w(Y) + \max_{Z \in \text{PREV}_X(Y; j)} \{\rho_{\text{odd}}^{*\uparrow}(X, Z; \lambda_A(Y) - 1)\} & \text{if } \text{PREV}_X(Y; j) \neq \emptyset, \\ -\infty & \text{otherwise,} \end{cases}$$

and for any $j \in [\lambda_A(Y) + 2, \gamma_A(Y) - 1]$,

$$\rho_{\text{odd}}^{*\uparrow}(X, Y; j) = \begin{cases} \max \left\{ \rho_{\text{odd}}^{*\uparrow}(X, Y; j - 1), w(Y) + \max_{Z \in \text{PREV}_X(Y; j)} \{\rho_{\text{odd}}^{*\uparrow}(X, Z; \lambda_A(Y) - 1)\} \right\} & \text{if } \text{PREV}_X(Y; j) \neq \emptyset, \\ \rho_{\text{odd}}^{*\uparrow}(X, Y; j - 1) & \text{otherwise.} \end{cases}$$

Given X , we can compute $\rho_{\text{odd}}^{*\uparrow}(X, Y)$ for all $Y \in \mathcal{X}$ in $O(k^2 + kn)$ time and space, as we have done for the even case. Consequently, we have the following lemma.

Lemma 14 *For a 2-layered edge-weighted bipartite graph G , a non-empty admissible set \mathcal{X} , we can compute a max-weighted odd-sized (X, Y) -path and its weight $\rho_{\text{odd}}^*(X, Y)$ for all $X, Y \in \mathcal{X}$ in $O(k^3 + k^2n)$ time and in $O(k^2 + kn)$ space.*

Construction of \mathcal{T} . Now we are ready to explain how to construct the collection \mathcal{T} of trapezoids.

Lemma 15 *Given an instance (G, w, \mathcal{X}) of the MW-2-CPEMP, we can construct a collection \mathcal{T} of weighted trapezoids in $O(k^3 + k^2n + m^2)$ time and in $O(k^2 + kn + m^2)$ space such that any optimal solution in the corresponding NTSP is also optimal for (G, w, \mathcal{X}) .*

PROOF: It suffices to collect all trapezoids that correspond to 2-CPE matchings M such that $H[M]$ (in the auxiliary graph H) is an isolated point, a cycle or a max-weighted (X, Y) -path for some $X, Y \in \mathcal{X}$. It takes $O(m)$ time for isolated points, $O(k^2 + m^2)$ time for cycles by Lemma 6, and $O(k^3 + k^2n)$ time for max-weighted (X, Y) -paths by Lemmas 12 and 14, where $\rho^*(X, Y) = \max\{\rho_{\text{odd}}^*(X, Y), \rho_{\text{even}}^*(X, Y)\}$ holds by (2). The space complexity is immediate from these lemmas. \square

This lemma tells $\Gamma = O(k^3 + k^2n + m^2)$. By Theorem 1 and $|\mathcal{T}| = O(k^2 + m)$, we have the following theorem immediately.

Theorem 4 *Given an instance (G, w, \mathcal{X}) of the MW-2-CPEMP, we can find a max-weighted 2-CPE matching in $O(k^3 + k^2n + m(m + \log n))$ time and in $O(k^2 + kn + m^2)$ space.*

5 Concluding Remarks

In the present paper, we have considered the max-weighted c -CPE matching problem (MW- c -CPEMP). Given (G, w, \mathcal{X}) , the problem asks to find a max-weighted matching M^* such that each edge in M^* intersects with at most c other edges in M^* , and that all edge crossings are contained in \mathcal{X} . The MW- c -CPEMP is regarded as a relaxation of the MW-0-CPEMP (i.e., max-weighted crossing-free matching problem). The degree of relaxation is represented by the constant c . Besides chronobiology, we believe that the MW- c -CPEMP should have many possibilities for application and extension.

Our approach reduces the MW- c -CPEMP to the non-contact trapezoid selection problem (NTSP). Given (\mathcal{T}, ω) , the NTSP asks to find a max-weighted subcollection of trapezoids such that no two of them contact each other. We propose an algorithm for the NTSP that runs in $O(|\mathcal{T}| \log n + n)$ time (Theorem 1). The proposed algorithm is an extension of the Malucelli et al.'s algorithm for the MW-0-CPEMP [10]. We construct an NTSP instance (\mathcal{T}, ω) such that any optimal solution for (\mathcal{T}, ω) is also optimal for the underlying MW- c -CPEMP instance. We explained how to construct such an NTSP instance and showed that the problem is polynomially solvable for $c \leq 2$ (Theorems 2 to 4). We leave analyses for the cases of $c \geq 3$ open.

The constraint that we have treated in the paper, at-most- c -crossings-per-edge, depends on the given drawing of the graph. We dealt with 2-layered drawing on parallel straight lines, one of the conventional drawing layouts, but the problem can be extended to others; e.g., 2-layered drawing on curves [8], 2-layered radial drawing [2], 2D geometry drawing [3]. Observe that the drawing implicitly specifies a conflict list of edge pairs that should not be included in a solution at the same time. Like the recent studies mentioned in Section 2.2, it would be interesting to explore problems of finding an “optimal” subgraph under the at-most- c -conflict constraint, where a conflict list is given as a part of the input instead of a graph drawing.

References

- [1] Altinel, I.K., Aras, N., Şuvak, Z., Taşkin, Z.C.: Minimum cost noncrossing flow problem on layered networks. *Discrete Applied Mathematics* (2018), in press
- [2] Battista, G.D., Eades, P., Tamassia, R., Tollis, I.G.: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall (1999)

- [3] Carlsson, J.G., Armbruster, B., Rahul, S., Bellam, H.: A bottleneck matching problem with edge-crossing constraints. *International Journal of Computational Geometry* 25(4), 245–261 (2015)
- [4] Chen, D.Z., Liu, X., Wang, H.: Computing maximum non-crossing matching in convex bipartite graphs. *Discrete Applied Mathematics* 187, 50–60 (2015)
- [5] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*, Third Edition, chap. 15.4. The MIT Press, 3rd edn. (2009)
- [6] Darmann, A., Pferschy, U., Schauer, J., Woeginger, G.J.: Path trees and matchings under disjunctive constraints. *Discrete Applied Mathematics* 159, 1726–1735 (2011)
- [7] Darmann, A., Pferschy, U., Schauer, J.: Determining a minimum spanning tree with disjunctive constraints. In: Rossi, F., Tsoukias, A. (eds.) *Algorithmic Decision Theory*. pp. 414–423. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
- [8] Giacomo, E.D., Grilli, L., Liotta, G.: Drawing bipartite graphs on two parallel convex curves. *Journal of Graph Algorithms and Applications* 12(1), 97–112 (2008)
- [9] Knauer, C., Schramm, E., Spillner, A., Wolff, A.: Configurations with few crossings in topological graphs. *Computational Geometry* 37, 104–114 (2007)
- [10] Malucelli, F., Ottmann, T., Pretolani, D.: Efficient labelling algorithms for the maximum noncrossing matching problem. *Discrete Applied Mathematics* 47, 175–179 (1993)
- [11] McCreight, E.: Priority search trees. *SIAM Journal on Scientific Computing* 14(2), 257–276 (1985)
- [12] Nishino, M., Suzuki, J., Umetani, S., Hirao, T., Nagata, M.: Sequence alignment as a set partitioning problem. *Journal of Natural Language Processing* 23(2), 175–194 (2016), written in Japanese
- [13] Öncan, T., Zhang, R., Punnen, A.P.: The minimum cost perfect matching problem with conflict pair constraints. *Computers & Operations Research* 40(4), 920–930 (2013)
- [14] Pferschy, U., Schauer, J.: The maximum flow problem with disjunctive constraints. *Journal of Combinatorial Optimization* 26(1), 109–119 (2013)
- [15] Ruangwises, S., Itoh, T.: Stable noncrossing matchings. CoRR abs/1903.02185 (2019), <http://arxiv.org/abs/1903.02185>
- [16] Samer, P., Urrutia, S.: A branch and cut algorithm for minimum spanning trees under conflict constraints. *Optimization Letters* 9(1), 41–55 (2015)
- [17] Setty, M., Tadmor, M.D., Reich-Zeliger, S., Angel, O., Salame, T.M., Kathail, P., Choi, K., Bendall, S., Friedman, N., Pe’er, D.: Wishbone identifies bifurcating developmental trajectories from single-cell data. *Nature Biotechnology* 34(6), 637–645 (2016)
- [18] Sugiyama, K.: *Graph Drawing and Applications: For Software and Knowledge Engineerings*, Series of Software Engineering and Knowledge Engineering, vol. 11. World Scientific (2002)
- [19] Sugiyama, K., Tagawa, S., Toda, M.: Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics* 11(2), 109–125 (1981)

- [20] Sun, M.: A tabu search heuristic procedure for solving the transportation problem with exclusionary side constraints. *Journal of Heuristics* 3, 305–326 (1998)
- [21] Sun, M.: The transportation problem with exclusionary side constraints and two branch-and-bound algorithms. *European Journal of Operational Research* 140, 620–647 (2002)
- [22] Torii, K., Inoue, K., Bekki, K., Haraguchi, K., Kubo, M., Kondo, Y., Suzuki, T., Shimizu, H., Uemoto, K., Saito, M., Fukuda, H., Araki, T., Endo, M.: Origination of the circadian clock system in stem cells regulates cell differentiation. *bioRxiv* (2019), <https://www.biorxiv.org/content/early/2019/07/22/710590>
- [23] Trapnell, C., Cacchiarelli, D., Grimsby, J., Pokharel, P., Li, S., Morse, M., Lennon, N.J., Livak, K.J., Mikkelsen, T.S., Rinn, J.L.: The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature Biotechnology* 32, 381–386 (2014)
- [24] Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. *Journal of Computational Biology* 1(4), 337–348 (1994)
- [25] Zhang, R., Kabadi, S.N., Punnen, A.P.: The minimum spanning tree problem with conflict constraints and its variations. *Discrete Optimization* 8(2), 191–205 (2011)