Connecting the Dots (with Minimum Crossings)

Akanksha Agrawal
Ben-Gurion University, Beer-Sheva, Israel
agrawal@post.bgu.ac.il

Grzegorz Guśpiel
Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland
guspiel@tcs.uj.edu.pl

Jayakrishnan Madathil
The Institute of Mathematical Sciences, HBNI, Chennai, India
jayakrishnanm@imsc.res.in

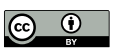
Saket Saurabh
The Institute of Mathematical Sciences, HBNI, Chennai, India
University of Bergen, Bergen, Norway
saket@imsc.res.in

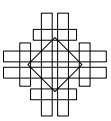
Meirav Zehavi
Ben-Gurion University, Beer-Sheva, Israel
meiravze@bgu.ac.il

Abstract

We study a prototype Crossing Minimization problem, defined as follows. Let $\mathcal{F}$ be an infinite family of (possibly vertex-labeled) graphs. Then, given a set P of (possibly labeled) n points in the Euclidean plane, a collection $L \subseteq \mathsf{Lines}(P) = \{\ell : \ell \text{ is a line segment with both endpoints in } P\}$, and a non-negative integer k, decide if there is a subcollection $L' \subseteq L$ such that the graph $G = (P, L')$ is isomorphic to a graph in $\mathcal{F}$ and L' has at most k crossings. By $G = (P, L')$, we refer to the graph on vertex set P, where two vertices are adjacent if and only if there is a line segment that connects them in L'. Intuitively, in Crossing Minimization, we have a set of locations of interest, and we want to build/draw/exhibit connections between them (where L indicates where it is feasible to have these connections) so that we obtain a structure in $\mathcal{F}$. Natural choices for $\mathcal{F}$ are the collections of perfect matchings, Hamiltonian paths, and graphs that contain an (s,t)-path (a path whose endpoints are labeled). While the objective of seeking a solution with few crossings is of interest from a theoretical point of view, it is also well motivated by a wide range of practical considerations. For example, links/roads (such as highways) may be cheaper to build and faster to traverse, and signals/moving objects would collide/interrupt each other less often. Further, graphs with fewer crossings are preferred for graphic user interfaces.

As a starting point for a systematic study, we consider a special case of Crossing Minimization. Already for this case, we obtain NP-hardness and W[1]-hardness results, and ETH-based lower bounds. Specifically, suppose that the input also contains a collection D of d non-crossing line segments such that each point in P belongs to exactly one line in D, and L does not contain line segments between points on the same line in D. Clearly, Crossing Minimization is the case where $d = n$ – then, P is in general position. The case of $d = 2$ is of interest not only because it is the most restricted non-trivial case, but also since it corresponds to a class of graphs that has been well studied – specifically, it is Crossing Minimization where $G = (P, L)$ is a (bipartite) graph with a so called *two-layer drawing*. For $d = 2$, we consider three basic choices of $\mathcal{F}$. For perfect matchings, we show *(i)* NP-hardness with an ETH-based lower bound, *(ii)* solvability in subexponential parameterized time, and *(iii)* existence of an $\mathcal{O}(k^2)$-vertex kernel. Second, for Hamiltonian paths, we show *(i)* solvability in subexponential parameterized time, and *(ii)* existence of an $\mathcal{O}(k^2)$-vertex kernel. Lastly, for graphs that contain an (s,t)-path, we show *(i)* NP-hardness and W[1]-hardness, and *(ii)* membership in XP.





35th International Symposium on Computational Geometry (SoCG 2019).
Editors: Gill Barequet and Yusu Wang; Article No. 7; pp. 7:1–7:17

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases crossing minimization, parameterized complexity, FPT algorithm, polynomial kernel, W[1]-hardness

Digital Object Identifier 10.4230/LIPIcs.SoCG.2019.7

Related Version A full version of the paper is available at `https://akanksha-agrawal.weebly.com/uploads/1/2/2/2/122276497/crossings.pdf`.

Funding *Akanksha Agrawal*: The work was carried out when the author was employed at Hungarian Academy of Sciences, and was supported by ERC Consolidator Grant SYSTEMATICGRAPH (No. 46 725978).
Grzegorz Guśpiel: Partially supported by the MNiSW grant DI2013 000443.
Saket Saurabh: Supported by ERC Consolidator Grant LOPPRE (No. 819416).
Meirav Zehavi: Supported by ISF grant no. 1176/18.

Acknowledgements We thank Grzegorz Gutowski and Paweł Rzążewski for many valuable comments regarding the NP-hardness proof for CM-PM.

1 Introduction

Let $\mathcal{F}$ be an infinite family of (possibly vertex-labeled) graphs. Suppose that given a graph F, the membership of F in $\mathcal{F}$ is testable in time polynomial in the size of F. For the family $\mathcal{F}$, we define a prototype Crossing Minimization problem as follows (see Fig. 1). Given a set P of (possibly labeled) n points in the two-dimensional Euclidean plane, a collection $L \subseteq \mathsf{Lines}(P) = \{\ell : \ell \text{ is a line segment with both endpoints in } P\}$, and a non-negative integer k, decide if there exists a subcollection $L' \subseteq L$ such that the graph $G = (P, L')$ is isomorphic[1] to a graph in $\mathcal{F}$ and L' has at most k crossings. The notation $G = (P, L')$ refers to the graph on vertex set P, where two vertices are adjacent if and only if there is a line segment that connects them in L'. Moreover, the number of crossings of L' is the number of pairs of line segments in L' that intersect each other at a point other than their possible common endpoint. The Crossing Minimization problem is a general model for a wide range of scenarios where we have a set of points of interest that correspond to geographical areas or fixed objects such as cities, manufacturing machinery or immobile equipment, attractions and mailboxes, and we want to build, draw or exhibit connections between them (where L indicates where it is feasible to have these connections) in order to obtain a structure in $\mathcal{F}$.

While the objective of seeking a solution with few crossings is of interest from a theoretical viewpoint, it is also well motivated by practical considerations. For example, public tracks (such as roads, highways or even paths in amusement parks) with fewer crossings require the construction of less bridges, elevated tracks, traffic lights and roundabouts, and therefore they are likely to be cheaper to build [42], easier and faster to traverse [10], and cause less accidents [21]. Moreover, signals and moving objects would interrupt each other less often. This property may be crucial as frequent collision between signals can distort or weaken them [3]. Furthermore, for moving objects such as robots (cleaning robots, autonomous agents and self-driving cars) that cannot physically be present in an intersection point simultaneously, encountering a large number of crossings may require the development of more complex navigation and sensory systems [37]. Lastly, graphs with fewer crossings are easier to view and analyze – in graphic user interfaces, for example, visual clarity is a major issue [13].

[1] With respect to vertex-labeled graphs, isomorphism also preserves the labeling of vertices rather than only their adjacency relationships – that is, a vertex labeled i can only be mapped to a vertex labeled i.

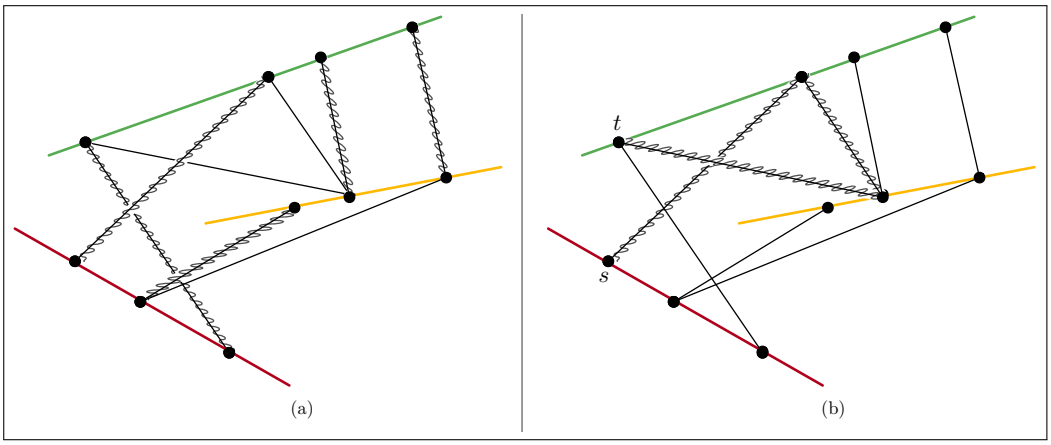


Figure 1 An instance of Crossing Minimization (in black) where $\mathcal{F}$ is the family of **(a)** perfect matchings, and **(b)** graphs that have an (s,t)-path. Solution edges are marked by squiggly lines – the number of crossings is 2 in **(a)** and 1 in **(b)**. The $d = 3$ colorful line segments display D.

Keeping the above applications in mind, three natural choices for the family $\mathcal{F}$ are the family of (Hamiltonian) paths, the family of graphs that contain an (s,t)-path (identification of s and t is modeled by vertex labels), and the family of (possibly vertex-labeled) perfect matchings. Indeed, these families model the most basic scenarios where all points must be connected by a path (e.g., to plan tracks for sightseeing trains or maintenance equipment such as cleaning robots or lawn mowers), only a specific pair of points must be connected by a path (e.g., to transport goods between two destinations), or the points are to be matched with one another (e.g., to pair up robots and charging ports). Furthermore, the computational problems that correspond to these families – Hamiltonian Path, (s,t)-Path and Perfect Matching, respectively – are among the most classical problems in computer science [22, 29, 18, 11].

As a starting point for a systematic study, we consider a special case of Crossing Minimization. Already for this case, we obtain NP-hardness and W[1]-hardness results, and ETH-based lower bounds, alongside positive results. Specifically, suppose that the input also contains a collection D of d non-crossing line segments such that each point in P belongs to exactly one line in D, and L does not contain line segments between points on the same line in D (see Fig. 1).[2] Clearly, Crossing Minimization is the case where $d = n$ – then, the set P can be in general position. The case of $d = 2$ is of interest not only because it is the most restricted non-trivial case, but also since it corresponds to a class of graphs that has been well studied in the literature – specifically, this case is precisely Crossing Minimization where $G = (P, L)$ is a (bipartite) graph with a so called *two-layer drawing*. Clearly, our hardness results carry over to any generalization of the case where $d = 2$. For this case, we consider the aforementioned three basic choices of $\mathcal{F}$, and obtain a comprehensive picture of their complexity. In what follows, we discuss our contribution, and then review related literature.

1.1 Our Contribution

Our study focuses on the class of two-layered graphs. Formally, a *two-layered graph* is a bipartite graph G with vertex bipartition $V(G) = X \cup Y$ that has a *two-layer drawing* – that is, a placement of the vertices of X on distinct points on a straight line segment L_1, and the

[2] Having lines segments between points on the same line in D only makes the problem more general.

vertices of Y on distinct points on a different straight line segment L_2 such that L_1 and L_2 are parallel to each other. (For ease of understanding, we take L_1 to be a segment of the line $y = 1$ in the plane, and similarly, L_2 to be a segment of the line $y = 0$.) The relative positions of the vertices in X and Y on L_1 and L_2, respectively, are given by permutations σ_X and σ_Y. Each edge is drawn using a straight line segment connecting the points of its end-vertices. We refer to (σ_X, σ_Y) as the *two-layered embedding/drawing* of G. Note that (σ_X, σ_Y) uniquely determines which edges intersect. The crossing minimization problem that corresponds to Perfect Matching on two-layered graphs is defined as follows.

Crossing-Minimizing Perfect Matching (CM-PM) **Parameter:** k
Input: A two-layered graph G (i.e., a bipartite graph G with bipartition $V(G) = X \cup Y$, and orderings σ_X and σ_Y of X and Y, respectively), and a non-negative integer k.
Question: Does G have a perfect matching with at most k crossings?

Similarly, we define the crossing minimization variants of Hamiltonian Path (the existence of a path that visits all vertices)[3] and (s,t)-Path (the existence of a path between two designated vertices). We refer to these problems as Crossing-Minimizing Hamiltonian Path (CM-HP) and Crossing-Minimizing (s,t)-Path (CM-Path).

Our Results. In this paper, we present a comprehensive picture of both the classical and parameterized computational complexities of these three problems as follows.[4]

CM-PM.

- **Negative.** NP-complete *even on graphs of maximum degree* 2. Moreover, unless the ETH fails, it can be solved neither in time $2^{o(n+m)}$ nor in time $2^{o(\sqrt{k})}n^{\mathcal{O}(1)}$ on these graphs, (where n and m are respectively the number of vertices and edges of the input graph.)
- **Positive.** Admits a kernel with $\mathcal{O}(k^2)$ vertices. Moreover, it admits a subexponential parameterized algorithm with running time $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$. In light of the negative result above, the running time of this algorithm is optimal under ETH.

We briefly remark that the proof of NP-completeness of CM-PM resolves an open question related to a problem called Token Swapping (see Section 1.2), introduced in 2014 by Yamanaka et al. [48, 49]. Two generalizations of Token Swapping were introduced by Yamanaka et al. [48, 49] and Bonnet et al. [7], both known to be NP-complete due to Miltzow et al. [40]. One of the results of Bonnet et al. [7] is the analysis of the complexity of all three token swapping problems on simple graph classes, including trees, cliques, stars and paths. Subset Token Swapping was shown to be NP-complete on the first three classes, but the status of the problem for paths was unknown. Since Subset Token Swapping restricted to paths is equivalent to our CM-PM (noted by Miltzow [39]), we derive that Subset Token Swapping restricted to paths is NP-complete as well.

[3] We remark that our results for Hamiltonian Path can be extended to Hamiltonian Cycle.
[4] Due to lack of space, several proofs have been omitted from the extended abstract.

CM-HP.

- **Negative.** NP-complete even on graphs that admit a Hamiltonian path. Unless the ETH fails, it can be solved neither in time $2^{o(n+m)}$ nor in time $2^{o(\sqrt{k})}n^{\mathcal{O}(1)}$ on these graphs.
- **Positive.** Admits a kernel with $\mathcal{O}(k^2)$ vertices. Moreover, it admits a subexponential parameterized algorithm with running time $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$. In light of the negative result above, the running time of this algorithm is almost optimal under ETH.

While Hamiltonian Path is a classical NP-complete problem [22], we prove that in the case of CM-HP, the hardness holds even if we know of a Hamiltonian path in the input graph (in which case Hamiltonian Path is trivial). We also comment that in the case of CM-HP (and also CM-Path), unlike the case of CM-PM, the problem becomes trivially solvable in polynomial time on graphs of maximum degree 2. Indeed, graphs of maximum degree 2 are collections of paths and cycles, and hence admit only linearly in n many Hamiltonian paths that can be easily enumerated in polynomial time. Then, CM-HP is solved by testing whether at least one of these Hamiltonian paths has at most k crossings. In fact, most natural NP-complete graph problems become solvable in polynomial time on graphs of maximum degree 2, therefore we find the hardness of CM-PM on these graphs quite surprising.

CM-Path.

- **Negative.** NP-complete and W[1]-hard. Specifically, unless W[1] = FPT, it admits neither an algorithm with running time $f(k)n^{\mathcal{O}(1)}$ nor a kernel of size $f(k)$, *for any computable function f of k.*
- **Positive.** Member in XP. Specifically, it is solvable in time $n^{\mathcal{O}(k)}$.

In light of our first two sets of results, we find our third set of results quite surprising: (s,t)-Path is the easiest to solve among itself, Perfect Matching and Hamiltonian Path,[5] yet when crossing minimization is involved, (s,t)-Path is substantially more difficult than the other two problems – indeed, CM-PM is not even FPT (unless W[1] = FPT).

Our Methods. In what follows, we give a brief overview of our methods.

CM-PM. We prove that CM-PM on graphs of maximum degree 2 is NP-hard by a reduction from Vertex Cover. The same reduction shows that CM-PM does not admit any $2^{o(n+m)}$-time (or $2^{o(\sqrt{k})}n^{\mathcal{O}(1)}$-time) algorithm unless the ETH fails.

For our algorithm and kernel, consider an instance (G,k) of CM-PM, where $V(G) = X \cup Y$ is the vertex bipartition with $|X| = |Y| = n$. For $i \in [n]$, let x_i and y_i denote the i^{th} vertices of X and Y, respectively, in the given two-layered embedding of G. It is not difficult to see that the only perfect matching with no crossings, if such a matching exists, is $\{x_iy_i \mid i \in [n]\}$. Therefore, if M is a perfect matching and $x_iy_j \in M$ with $i \neq j$, then the edge x_iy_i must

[5] In particular, (s,t)-Path can be directly solved in linear time via BFS [11], while Perfect Matching is only known to be solvable by more complex (non-linear time) algorithms such as Edmonds algorithm [18], and the status of Hamiltonian Path is even worse given that it is NP-complete [22].

intersect another edge in M, which yields a crossing. In fact, x_iy_j must intersect at least $|j-i|$ edges. Therefore, no feasible solution for CM-PM can contain an edge x_iy_j with $|j-i| > k$. This observation plays a key role in both our algorithm and kernel designs. Our algorithm is based on dynamic programming (DP), and its analysis is based on Hardy-Ramanujan numbers [26]. (By considering these numbers, we are able to derive a running time bound of $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k})})$.) Very briefly, at stage i we consider the graph G_i, the subgraph of G induced by $X_i \cup Y_i = \{x_j, y_j \mid j \leq i\}$. Our algorithm "guesses" which subsets of $V(G_i)$ are going to be matched to "future vertices", i.e., vertices in $V(G) \setminus V(G_i)$, in an optimal solution, and solves the problem optimally on the graph induced by the remaining vertices. For the kernel, we show that either (G, k) is a no-instance or the number of "bad pairs", i.e., $\{x_i, y_i\}$ where $x_iy_i \notin E(G)$, cannot exceed $2k$. We then bound the number of pairs $\{x_i, y_i\}$ between two consecutive bad pairs by $\mathcal{O}(k)$ again, which gives a kernel with $\mathcal{O}(k^2)$ vertices.

CM-HP. By a reduction from a variant of Hamiltonian Path on bipartite graphs, we show that CM-HP is NP-hard even if the input graph is assumed to have a Hamiltonian path. For our FPT algorithm and kernel, we adopt a strategy similar to the one we employed for CM-PM.

CM-PATH. We prove the W[1]-hardness of CM-Path by giving an appropriate reduction from Multi-Colored Clique, which is known to be W[1]-hard [19]. Given an instance $(G, V_1, V_2, \ldots, V_k)$ of Multi-Colored Clique (G is a k-partite graph, and the problem is to check whether G contains a clique with exactly one vertex from each V_i), we create an equivalent instance (G', X, Y, s, t, k') of CM-Path, where G' is a two-layered graph, as follows. We create an s-t path in G' that "selects" a vertex from each V_i and an edge for each (distinct) pair (V_i, V_j). To this end, for each V_i, we have a vertex selection gadget $\mathcal{V}_i$, and for (distinct) V_i, V_j, we have an edge selection gadget $\mathcal{E}_{ij}$. The vertex and edge selection gadgets are arranged in a linear fashion to create an $s-t$ path in G'. In the construction, we add a pair of non-adjacent vertices in $\mathcal{E}_{ij}$ for each edge between V_i and V_j. We also add a path between the pair of (non-adjacent) vertices whose edges cross the gadgets $\mathcal{V}_i$ and $\mathcal{V}_j$, which enforces compatibility between vertices and edges that are selected. Finally, by setting k' appropriately, we get the desired reduction.

As for the XP algorithm for CM-Path, we guess which edges of G are going to be involved in crossings in a feasible solution. The problem then reduces to connecting these guessed edges using crossing-free subpaths, which can be done in polynomial time.

1.2 Related Works

The Crossing Number Problem. The *crossing number* of a graph G is the minimum number of crossings in a plane drawing of G. The notion of a crossing number originally arose in 1940 by Turán [46] for bipartite graphs in the context of the minimization of the number of crossings between tracks connecting brick kilns to storage sites. Computationally, the input of the Crossing Number problem is a graph G and a non-negative integer k, and the task is to decide whether the crossing number of G is at most k. This problem is among the most classical and fundamental graph layout problems in computer science. It was shown to be NP-complete by Garey and Johnson in 1983 [23]. Not only is the problem NP-complete on graphs of maximum degree 3 [27], but also it is surprisingly NP-complete even on graphs that can be made planar and hence crossing-free by the removal of just a single edge [8]. Nevertheless, Crossing Number was shown to be FPT by Grohe already in 2001 [24], who

developed an algorithm that runs in time $f(k)n^2$ where f is at least double exponential.[6] A further development was achieved by Kawarabayashi and Reed [32], who showed that the problem is solvable in time $f(k)n$. On the negative side, Hlinený and Dernár [28] proved that Crossing Number does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.

Variants of Crossing Number where the vertices can be placed only on prespecified curves are extensively studied. Closely related to our work is the well-known Two-Layer Crossing Minimization problem: given a bipartite graph G with vertex bipartition $V(G) = X \cup Y$, and a non-negative integer k, the task is to decide whether G admits a two-layered drawing where the number of crossings is at most k. This problem originated in VLSI design [44]. A solution to the Two-Layer Crossing Minimization problem is also useful in solving the rank aggregation problem, which has applications in meta-search and spam reduction on the Web [6]. We refer the reader to [50] and references therein for other applications. The Two-Layer Crossing Minimization problem is long known to be NP-complete, even in its one sided version where we are allowed to permute vertices only from one (fixed) side [16, 17]. Further, the membership of Two-Layer Crossing Minimization in FPT has already been proven close to two decades ago by Dujmovic et al. [15]. Noteworthy is also the well-studied variant of Crossing Number that restricts the vertices to be placed only on a prespecified circle and edges are drawn as straight line segments. Both of these variants as well as their various versions are subject to an active line of research [33]. Further, aesthetic display of these layouts are of importance in biology [36], and included in standard graph layout software [31] such as yFiles, Graphviz, or OGDF. For more information on Crossing Number and its variants, we refer to surveys such as [43].

Problems on Fixed Point Sets. Settings where we are given a set P of points in the plane that represent vertices, and edges are to be drawn as straight lines between them, are intensively studied since the early 80s. A large body of work has been devoted to the establishment of combinatorial bounds on the number of *crossing-free* graphs on P, where particular attention is given to crossing-free triangulations, perfect matchings and Hamiltonian paths and cycles. Originally, the study of these bounds was initiated by Newborn and Moser in 1980 [41] for crossing-free Hamiltonian cycles. For more information, we refer to the excellent Introduction of Sharir and Welzl [45] and the references therein. Computationally, the problem of *counting* the number of such crossing-free graphs (faster than the time required to enumerate them) is of great interest (see, e.g., [47, 4, 38]). Furthermore, the computation of a single crossing-free graph on P (such as a perfect matching), possibly with a special property of being "short" [2, 1, 9], has already been studied since 1993 [30]. To the best of our knowledge, the minimization of the number of crossings (rather than the detection of a crossing-free graph) has received only little attention, mostly in an ad-hoc fashion. An exception to this is the work of Halldórsson et al. [25] with respect to spanning trees. We remark that they study the problem in its full generality, where the computation of even a crossing-free spanning tree is already NP-complete [34, 30].

Related to our study is also the Metro Line Crossing Minimization problem, introduced by Benkert et al. [5]. Given an embedded graph G on P, as well as k pairs of vertices (called terminals), a solution to this problem is a set of paths that connect their respective pairs of terminals, and which has minimum number of "crossings" under a definition different

[6] We find the contrast between this result and our result on CM-Path somewhat surprising. At first glance, our CM-Path problem seems computationally simpler than Crossing Number (where the embedding is computed from scratch), yet our problem is W[1]-hard while Crossing Number is FPT.

than ours. Specifically, paths are thought of as being drawn in the plane "alongside" the edges of G rather than on the edges themselves. Such a formulation allows to reuse a single edge a large number of times. Therefore, the avoidance of crossings might come at the cost of congesting the same tracks by buses and trains (or building many parallel tracks).

2 Preliminaries

We use $\mathbb{N}$ to denote $\{0, 1, 2, \ldots\}$. For $n \in \mathbb{N}$, let $[n] = \{1, 2, 3, \ldots, n\}$, and $[n]_0 = [n] \cup \{0\}$.

Two-layered graphs. Whenever context is clear, denote the vertex bipartition of a two-layered graph G (given by the two-layer drawing) by (X, Y). Let $n_X = |X|$ and $n_Y = |Y|$. For $i \in [n_X]$, let x_i be the ith vertex of X, and for $j \in [n_Y]$, let y_j be the jth vertex of Y. Moreover, we say that i is the index of the vertex x_i, and j is the index of the vertex y_j; we write $\texttt{index}(x_i) = i$ and $\texttt{index}(y_j) = j$. Similarly, let X_i denote $\{x_r \mid 1 \leq r \leq i\}$, and let Y_j denote $\{y_r \mid 1 \leq r \leq j\}$. For $i, j \in [n_X]$, where $i \leq j$, the set $X_{i,j}$ denotes the set $\{x_p \mid i \leq p \leq j\}$. Moreover, if $i < j$, then $X_{j,i} = \emptyset$. The set $Y_{i,j}$ is defined analogously for $i, j \in [n_Y]$. A *crossing* in G is a pair of edges intersecting at a point other than their possible common endpoints. Note that two edges $x_i y_j$ and $x_r y_s$, where $i, r \in [n_X]$ and $j, s \in [n_Y]$, form a crossing (or, cross each other) if and only if $r > i, j > s$ or $i > r, s > j$. For a subgraph H of G, $\texttt{cr}(H)$ denotes the number of crossings in H. Similarly, for a set of edges $E' \subseteq E(G)$, $\texttt{cr}(E')$ denotes the number of crossings in the subgraph induced by E'.

For an introduction to parameterized complexity and kernelization, see [12, 14, 20].

3 NP-hardness for CM-PM

We show that CM-PM is NP-hard via a reduction from Vertex Cover (known to be NP-hard from [35]). The Vertex Cover problem takes as input a graph H and an integer l, and the goal is to check if there is $S \subseteq V(H)$ of size at most l, such that $H - S$ has no edges.

Let (H, l) be an instance of Vertex Cover. We create a corresponding instance (G, k) of CM-PM. The general idea behind the reduction is to use two gadgets. The first one is created for every vertex of H. There are two possible perfect matchings in each copy of the gadget. Selecting one of these matchigs corresponds to choosing whether the vertex belongs to the vertex cover or not. The second gadget is created for every edge of H. There are also two possible perfect matchings in each copy of the gadget, corresponding to orienting the edge in one of the two possible directions. The details of the construction ensure that the number of crossings is minimized when each edge is oriented towards a selected vertex and the number of selected vertices is minimal. We assume without loss of generality that the two straight line segments are parallel and horizontal. Vertex gadgets are aligned in such a way that there are no crossings between them, but each of them defines regions between the vertices, called slots, that are used to anchor edge gadgets. The heart of the argument is a careful analysis of the number of crossings between different gadgets.

For any integer $s \geqslant 1$, the *vertex gadget of size s* is a cycle on $8s$ vertices together with a path on 2 vertices, positioned as shown in Figure 2. The vertex gadget defines $2s$ *slots*. The slots are spaces between the vertices, whose exact location is marked in Figure 2 using gray rectangles. The ones to the left of the pink edge are called *left slots* and the ones to the right are called *right slots*. Furthermore, observe that there are only two ways to choose a perfect matching in this gadget: either take all the blue edges and the pink edge in the middle, or take all the yellow edges and the pink one. We interpret choosing the blue (yellow) matching as selecting (not selecting) the vertex to the vertex cover.

Figure 2 The vertex gadget of size 4, with 8 slots colored gray.

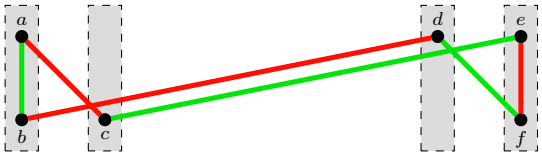


Figure 3 The edge gadget and the placement of its vertices in slots.

We fix an ordering $v_1, \ldots, v_n$ on the vertices of H. For every $v \in V(H)$, we create a copy of the vertex gadget of size $2d(v)$. We arrange the gadgets on the two line segments in such a way that each gadget occupies a separate range of the x axis and for every $i < j$, the gadget for v_i is to the left of the gadget for v_j.

We process the edges in any order. For every $v_iv_j \in E(H)$, where $i < j$, we select two first unselected right slots in the gadget for v_i and two first unselected left slots in the gadget for v_j. The *edge gadget* for v_iv_j is a cycle on 6 vertices that are labeled and arranged on the line segments as shown in Figure 3. These vertices are carefully placed in the slots as follows: vertices a and b in the first of the two selected slots of the gadget for v_i, vertex c in the second of these two slots, vertex d in the first of the two selected slots of the gadget for v_j, and vertices e and f in the second of these two slots. The edge gadget admits two different perfect matchings. We interpret choosing the green (red) matching as orienting the edge towards v_i (v_j). For a complete example of the reduction for a small graph, see Figure 4.

Now by setting the "budget" for the number of edge crossings appropriately, we can obtain the following theorem (we refer to the full version for a detailed analysis).

▶ **Theorem 1.** CM-PM *is NP-hard, even if the maximum degree of the input graph is 2.*

4 FPT Algorithm for CM-PM

Let (G, k) be an instance of CM-PM, with vertex bipartition X and Y, where $|X| = |Y| = n$. (Here, we note that if $|X| \neq |Y|$ then (G, k) is a no-instance as it does not admit a perfect matching.) We will design an FPT algorithm for CM-PM running in time $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$. Our algorithm will be a DP algorithm which processes the graph from left to right. That is to say, for each $i = 1, 2, \ldots, n$, at stage i, we consider the graph $G_i = G[X_i \cup Y_i]$, the graph induced by $\{x_1, \ldots, x_i, y_1, \ldots, y_i\}$, and solve a family of subproblems, the solution for one of which will lead to an optimal solution for the entire graph G. We will bound the number of sub-instances that we need to solve at each stage i, for $i \in [n]$, by $2^{\mathcal{O}(\sqrt{k})}$. To achieve the above, we will use a well-known result on the number of partitions of an integer, (which says that the number of partitions of an integer k is $2^{\mathcal{O}(\sqrt{(k)})}$). (For the integer 6, a partition of it is $1 + 2 + 3$.) We will rely on the fact that for a number t, we can compute all its partitions in time bounded by $2^{\mathcal{O}(\sqrt{t})}$. This bound will be crucial for achieving the running time.

We first explain the intuition behind our algorithm. Suppose (G, k) is a yes-instance and let M be a perfect matching of G with $\mathtt{cr}(M) \leqslant k$. Fix $i \in [n]$. Consider how M saturates the "future vertices," i.e., vertices in $X_{i+1,n} \cup Y_{i+1,n}$. Consider a future vertex, say x_j for some $j > i$. Using the fact that $\mathtt{cr}(M) \leqslant k$, we will show that M cannot match x_j to a vertex

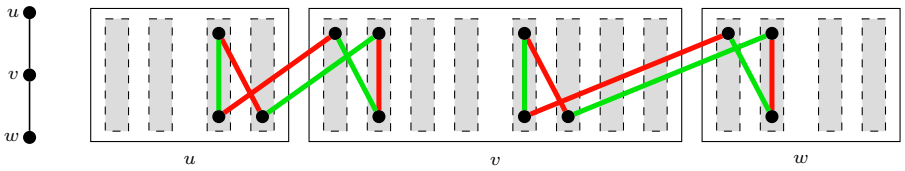


Figure 4 A graph H and the two-layered graph G obtained by passing H to the reduction algorithm, vertex gadgets presented schematically.

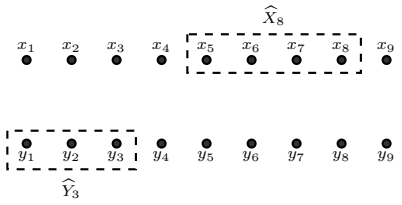
in Y_{i-k}. Therefore, the only vertices in $X_i \cup Y_i$ that can possibly be matched to vertices in the future belong to $X_{i-k+1} \cup Y_{i-k+1}$. In other words, while doing a DP from left to right, by the time we get to stage i, the intersection of the potential solution with $X_{i-k} \cup Y_{i-k}$ is completely determined. This observation suggests the most obvious strategy: at stage i, "guess" how the solution matches (and saturates) the vertices in $X_{i-k+1,i} \cup Y_{i-k+1,i}$. But this strategy will only lead to an algorithm running in time $k^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$. Observe that since we are only interested in a matching with the least possible number of crossings, we need not look at all possible matchings in $G[X_{i-k+1,i} \cup Y_{i-k+1,i}]$. We only need to look at which subsets of $X_{i-k+1,i}$ and $Y_{i-k+1,i}$ are saturated by M. Thus, from each collection of matchings that saturate the same subset of $X_{i-k+1,i} \cup Y_{i-k+1,i}$, we remember the matching that incurs the least number of crossings. This observation can be used to obtain an algorithm running in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$. To further improve this running time, we show that the number of subsets of $X_{i-k+1,i} \cup Y_{i-k+1,i}$ that are not saturated by the intersection of any potential solution with $X_i \cup Y_i$ cannot exceed $2^{\mathcal{O}(\sqrt{k})}$. (This is where we will use the bound that the number of partitions of an integer t is bounded by $2^{\mathcal{O}(\sqrt{t})}$.) This will lead us to an algorithm with the claimed running time for the problem.

We start by giving some notations and preliminary results that will be helpful later. We let $\mathsf{Sat}(M) = \{u, v \mid uv \in M\}$. That is, $\mathsf{Sat}(M)$ is the set of vertices saturated by M in G.

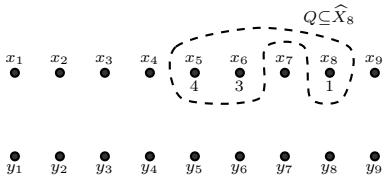
Partitions of an integer. For a positive integer α, a partition of α refers to writing α as a sum of positive integers (greater than zero), where the order of the summands is immaterial. Each summand in such a sum is called a *part* of α. For example, $16 = 1 + 4 + 4 + 7$ is a partition of 16. Note that here two of the parts (the two 4s) are the same. We, however, are interested in only those partitions of α in which the parts are all distinct. Let us call such partitions *distinct-part partitions*. For example, $\{1, 2, 6, 7\}$ is a distinct-part partition of 16. It is known that the number of partitions (and hence the number of distinct-part partitions) of an integer k is bounded by $2^{\mathcal{O}(\sqrt{k})}$ [26]. In light of this result, it is not difficult to see that given an integer k, all distinct-part partitions of k can be generated in time $2^{\mathcal{O}(\sqrt{k})}$. For future reference, we state these results below.

▶ **Lemma 2.** *The number of distinct-part partitions of any positive integer k, is at most $2^{\mathcal{O}(\sqrt{k})}$. Moreover, we can generate all of these distinct-part partitions in time $2^{\mathcal{O}(\sqrt{k})}$.*

Some important sets for the algorithm. For $i \in [n]$, we let $\widehat{X}_i = \{x_{i-k+\ell} \mid \ell \in [k] \text{ and } i - k + \ell \geqslant 1\}$ and $\widehat{Y}_i = \{y_{i-k+\ell} \mid \ell \in [k] \text{ and } i - k + \ell \geqslant 1\}$ (see Figure 5). We will argue that in any perfect matching M in G with $\mathtt{cr}(M) \leqslant k$, the vertices from X_i which are matched to a vertex y_s, with $s \geqslant i + 1$, belong to the set $\widehat{X}_i$. Similarly, we can argue that $\widehat{Y}_i$ is the set of vertices from Y_i which can possibly be matched to vertices x_s, with $s \geqslant i + 1$.



(a) The dashed rectangles show $\widehat{Y}_3$ and $\widehat{X}_8$, when $k = 4$.



(b) $k = 4$. $Q = \{x_5, x_6, x_8\} \subseteq \widehat{X}_8$. The number below each $x_j \in Q$ shows $\mathsf{cst}_8(x_j)$. $\mathsf{CstSet}_8(Q) = \{4, 3, 1\}$ and $\mathsf{cst}_8(Q) = 8$.

Figure 5 An example of $\widehat{X}_i, \widehat{Y}_i, Q \subseteq \widehat{X}_i, \mathsf{CstSet}_i(Q)$ and $\mathsf{cst}_i(Q)$.

We will now associate costs to vertices (and subsets) of $\widehat{X}_i$ (resp. $\widehat{Y}_i$), which will be helpful in obtaining lower bounds on the number of crossings, when vertices from $\widehat{X}_i$ (resp. $\widehat{Y}_i$) are matched to vertices y_s (resp. x_s), where $s \geqslant i+1$. To this end, consider $i \in [n]$ and a vertex $x_r \in \widehat{X}_i$. We let $\mathsf{cst}_i(x_r) = i + 1 - r$. Since $x_r \in \widehat{X}_i$, we have $r \leqslant i$, and thus, $\mathsf{cst}_i(x_r) \geqslant 1$. For a subset $Q \subseteq \widehat{X}_i$, we let $\mathsf{CstSet}_i(Q) = \{\mathsf{cst}_i(x) \mid x \in Q\}$ and $\mathsf{cst}_i(Q) = \sum_{x \in Q} \mathsf{cst}_i(x)$. Similarly, for $i \in [n]$ and a vertex $y_r \in \widehat{Y}_i$, we let $\mathsf{cst}_i(y_r) = i + 1 - r \geqslant 1$. Moreover, for a subset $Q \subseteq \widehat{Y}_i$, we let $\mathsf{CstSet}_i(Q) = \{\mathsf{cst}_i(y) \mid y \in Q\}$ and $\mathsf{cst}_i(Q) = \sum_{y \in Q} \mathsf{cst}_i(y)$. We note that, for each $i \in [n]$, we have $\mathsf{cst}_i(\emptyset) = 0$. In order to understand the intuition behind these definitions, look at the ith stage in our algorithm. At stage i, we consider the graph $G[X_i \cup Y_i]$. Consider the vertices in $\widehat{X}_i$ that are matched to vertices in the future (i.e., vertices y_s where $s > i$). Note that if x_i gets matched to a future vertex, then x_i participates in at least one crossing (in the final solution), and if x_{i-1} gets matched to a future vertex, then x_{i-1} participates in at least two crossings and so on. In particular, $x_r \in \widehat{X}_i$, if matched to a future vertex participates in at least $i+1-r$ crossings. So, $\mathsf{cst}_i(x_r)$ is a lower bound on the number of crossings in which x_r participates (or cost incurred by x_r) if it gets matched to a future vertex. For a set $Q \subseteq \widehat{X}_i$, $\mathsf{CstSet}_i(Q)$ is the set of minimum costs incurred by each element of Q. Moreover, $\mathsf{cst}_i(Q)$ is the cost incurred by Q if all its elements get matched to future vertices. Now using the notion of distinct-part partitions of an integer, we introduce some "special" sets of subsets of X and Y, respectively. These sets will be crucially used while creating the sub-instances in our DP algorithm. For $\alpha \in [k]$, let $\mathcal{P}_\alpha$ be the set of all distinct-part partitions of α. Furthermore, let $\mathcal{P}_{\leqslant k} = \cup_{\alpha \in [k]} \mathcal{P}_\alpha$. From Lemma 2, we have $|\mathcal{P}_{\leqslant k}| = 2^{\mathcal{O}(\sqrt{k})}$. Consider $i \in [n]$, $\alpha \in [k]$, and $\mathtt{P} \in \mathcal{P}_{\leqslant \alpha}$. We let $S^i_X(\mathtt{P}) = \{x_{i+1-\beta} \mid \beta \in \mathtt{P} \text{ and } i + 1 - \beta \geqslant 1\}$. (For example, for $\mathtt{P} = \{1, 2, 6, 7, 8\}$ and $i = 6$, we have $S^i_X(\mathtt{P}) = \{x_6, x_5, x_1\}$.) Note that $S^i_X(\mathtt{P}) \subseteq \widehat{X}_i$, $\mathsf{CstSet}_i(S^i_X(\mathtt{P})) = \mathtt{P}$, and $\mathsf{cst}_i(S^i_X(\mathtt{P})) = \alpha$, where $\mathtt{P}$ is a partition of $\alpha \in [k]$. Similarly, we define $S^i_Y(\mathtt{P}) = \{y_{i+1-\beta} \mid \beta \in \mathtt{P} \text{ and } i + 1 - \beta \geqslant 1\} \subseteq \widehat{Y}_i$. Again, note that $\mathsf{CstSet}_i(S^i_X(\mathtt{P})) = \mathtt{P}$ and $\mathsf{cst}_i(S^i_X(\mathtt{P})) = \alpha$. We let $\mathcal{S}^i_X = \{S^i_X(\mathtt{P}) \mid \mathtt{P} \in \mathcal{P}_{\leqslant k}\} \cup \{\emptyset\} \subseteq 2^{\widehat{X}_i}$ and $\mathcal{S}^i_Y = \{S^i_Y(\mathtt{P}) \mid \mathtt{P} \in \mathcal{P}_{\leqslant k}\} \cup \{\emptyset\} \subseteq 2^{\widehat{Y}_i}$.

▶ **Lemma 3.** *The families $\mathcal{S}^i_X$ and $\mathcal{S}^i_Y$ contain at most $|\mathcal{P}_{\leqslant k}| + 1 = 2^{\mathcal{O}(\sqrt{k})}$ sets each. Moreover, for each $i \in [n]$, the families $\mathcal{S}^i_X$ and $\mathcal{S}^i_Y$ can be generated in $2^{\mathcal{O}(\sqrt{k})}$ time.*

We associate a set of integers to every pair $(S, S') \in \mathcal{S}^i_X \times \mathcal{S}^i_Y$, for each $i \in [n]$. These sets will give the "allowed" number of crossings for a matching in the graph G_i. Consider $i \in [n]$, $S \in \mathcal{S}^i_X$, and $S' \in \mathcal{S}^i_Y$. We let $\mathsf{Alw}_i(S, S') = \{\ell \in [k]_0 \mid \ell \leqslant k - \max\{\mathsf{cst}_i(S), \mathsf{cst}_i(S')\}\}$.

▶ **Observation 4.** *Consider $i \in [n] \setminus \{1\}$. For $S \in \mathcal{S}^i_X$ and $Q \subseteq S \setminus \{x_i\}$, we have $Q \in \mathcal{S}^{i-1}_X$. Similarly, for $S' \in \mathcal{S}^i_Y$ and $Q' \subseteq S' \setminus \{y_i\}$, we have $Q' \in \mathcal{S}^{i-1}_Y$.*

▶ **Observation 5.** *Consider $i \in [n] \setminus \{1\}$. For $S \in \mathcal{S}^i_X$ and $Q \subseteq S \setminus \{x_i\}$, we have $\mathsf{cst}_{i-1}(Q) \leqslant \mathsf{cst}_i(S) - |S|$. Similarly, for $S' \in \mathcal{S}^i_Y$ and $Q' \subseteq S' \setminus \{x_i\}$, we have $\mathsf{cst}_{i-1}(Q') \leqslant \mathsf{cst}_i(S') - |S'|$.*

We now define the notion of a "compatible matching." Consider $i \in [n]$, $S \in \mathcal{S}^i_X$, and $S' \in \mathcal{S}^i_Y$. We say that a matching M in G_i is (i, S, S')-*compatible* if $S = \widehat{X}_i \setminus \mathsf{Sat}(M)$, $S' = \widehat{Y}_i \setminus \mathsf{Sat}(M)$, and $\mathtt{cr}(M) \leqslant k - \max\{\mathsf{cst}_i(S), \mathsf{cst}_i(S')\}$. Compatible matchings will be helpful in establishing the correctness of our algorithm, in which we will be considering matchings of G_i that saturate exactly $(X_i \cup Y_i) \setminus (S \cup S')$, while incurring at most a certain allowed number of crossings. Suppose at the ith stage of our algorithm, we consider a matching, say M_i, of G_i that does not saturate S. We would like to extend M_i to a matching of G with at most k crossings. That is, at stage i, M_i matches S to future vertices. Therefore, while extending M_i to a matching of the entire graph G, we will incur at least $\mathsf{cst}_i(S)$ more crossings (in addition to $\mathtt{cr}(M_i)$). Therefore, in order to be able to extend M_i to matching of G with at most k crossings, $\mathtt{cr}(M_i)$ cannot exceed $k - \mathsf{cst}_i(S)$. Identical reasoning holds for the set S'.

We are now ready to define the states of our DP table. For each $i \in [n]$, $S \in \mathcal{S}^i_X$ and $S' \in \mathcal{S}^i_Y$ with $|S| = |S'|$, and an integer $\ell \in \mathsf{Alw}_i(S, S') = \{\ell \in [k]_0 \mid \ell \leqslant k - \max\{\mathsf{cst}_i(S), \mathsf{cst}_i(S')\}\}$, we define

$$T[i, S, S', \ell] = \begin{cases} 1, & \text{if there is a matching } M \text{ in } G_i, \text{ such that } \mathtt{cr}(M) = \ell \text{ and} \\ & \mathsf{Sat}(M) = (X_i \setminus S) \cup (Y_i \setminus S'), \\ 0, & \text{otherwise.} \end{cases}$$

Observe that (G, k) is a yes-instance of CM-PM if and only if there is $\ell \in [k]_0$, such that $T[n, \emptyset, \emptyset, \ell] = 1$. A matching M in G_i is said to *realize* $T[i, S, S', \ell]$, if $\mathtt{cr}(M) = \ell$ and M is (i, S, S')-compatible. In the above we note that $\ell \leqslant k - \max\{\mathsf{cst}_i(S), \mathsf{cst}_i(S')\}$, as $\ell \in \mathsf{Alw}_i(S, S')$. Let us now see how $T[i, S, S', \ell]$ can be computed.

Base Case. Consider the entry $T[1, S, S', \ell]$. Note that $\mathtt{cr}(G_1) = 0$. Thus, if $\ell > 0$, we have $T[1, S, S', \ell] = 0$. Now we consider the case when $\ell = 0$. Recall that by definition, we have $|S| = |S'|$. If $S = \{x_1\}$ and $S' = \{y_1\}$, then we should not match any vertex. Thus, we have a matching (which is the empty set) with 0 crossings, and thus, $T[1, S, S', \ell] = 1$. Otherwise, we have $S = S' = \emptyset$. Note that the only possible matching in the graph $G[\{x_1, y_1\}]$ is $\{x_1y_1\}$. So, if $x_1y_1 \in E(G)$, then $\{x_1y_1\}$ is a matching with 0 crossings, and hence $T[1, S, S', \ell] = 0$. Otherwise, we have $x_1y_1 \notin E(G)$, and hence $T[1, S, S', \ell] = 0$.

We now move to our recursive formulae for the computation of the entries of our table. We set the value of $T[i, S, S', \ell]$ (recursively) based on the following cases, where $i > 1$.

Case 1: $x_i \in S$ and $y_i \in S'$. From Observation 4, we have that $S \setminus \{x_i\} \in \mathcal{S}^{i-1}_X$ and $S' \setminus \{y_i\} \in \mathcal{S}^{i-1}_Y$. Also, from Observation 5 it follows that $\ell \in \mathsf{Alw}_{i-1}(S \setminus \{x_i\}, S' \setminus \{y_i\})$. We set $T[i, S, S', \ell] = T[i-1, S \setminus \{x_i\}, S' \setminus \{y_i\}, \ell]$.

▶ **Lemma 6.** *The computation of $T[i, S, S', \ell]$ in Case 1 is correct.*

Case 2: $x_i \in S$ and $y_i \notin S'$, or $x_i \notin S$ and $y_i \in S'$. We will only argue for the case when $x_i \in S$ and $y_i \notin S'$. (The other case can be handled symmetrically.) Thus, hereafter we assume that $x_i \in S$ and $y_i \notin S'$. In this case, a matching, say M, which realizes $T[i, S, S', \ell]$, must saturate the vertex y_i and must not saturate the vertex x_i. Thus, M must have an edge x_jy_i, where $j < i$ (here we rely on the fact that y_i cannot be matched to x_i, as $x_i \in S$).

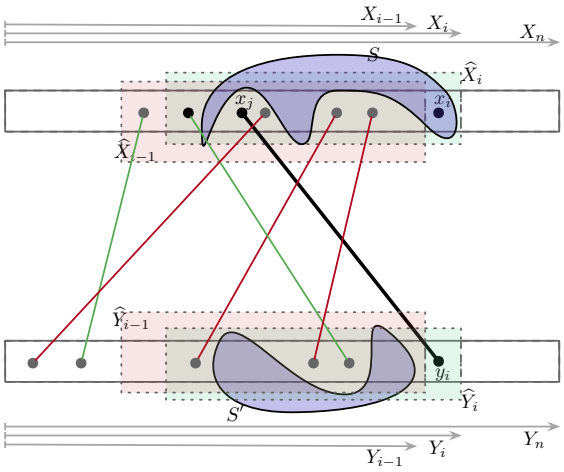


Figure 6 An illustration of the edges intersecting x_jy_i, where $x_j \in \widehat{X}_{i-1} \setminus S$. Here, the red edges intersect x_jy_i and the green edges do not intersect x_jy_i.

As M must satisfy the constraint $\mathtt{cr}(M) = \ell \leqslant k$, we must have $i - k \leqslant j < i$. That is, the vertex to which y_i is matched, must belong to the set $\widehat{X}_{i-1}$. We will construct a set $\mathcal{Q} \subseteq \mathcal{S}_X^{i-1} \subseteq 2^{\widehat{X}_{i-1}}$. This set will be used for creating sub-instances whose values are needed for the computation of $T[i, S, S', \ell]$. Intuitively speaking, each set in $\mathcal{Q}$ will determine a vertex to which y_i is matched, in the matching that we are seeking for. Note that as y_i must be saturated by any matching that realizes (or complies) with $T[i, S, S', \ell]$, the edge, say $\widehat{x}y_i$ in the matching might intersect other edges of the matching. Therefore, we will have to account for this extra overhead in the number of crossing edges. To count these extra crossings incurred, we will define an "overhead" function.

To construct $\mathcal{Q}$, we first construct two sets $\widehat{\mathcal{Q}}, \widetilde{\mathcal{Q}} \subseteq 2^{\widehat{X}_{i-1}}$ (each of size at most $\mathcal{O}(k)$). We will obtain $\widehat{\mathcal{Q}} \supseteq \widetilde{\mathcal{Q}} \supseteq \mathcal{Q}$ (in that order, by removing some "bad sets"). For a vertex $x_j \in (N(y_i) \cap \widehat{X}_{i-1}) \setminus S$, let $Q_j = (S \setminus \{x_i\}) \cup \{x_j\}$. Intuitively, the vertex y_i will be matched to x_j, when Q_j is under consideration. Note that $Q_j \subseteq \widehat{X}_{i-1}$. We let $\widehat{\mathcal{Q}} = \{Q_j \mid x_j \in (N(y_i) \cap \widehat{X}_{i-1}) \setminus S\}$. In the above definition, we only consider the neighbors of y_i from $\widehat{X}_{i-1} \setminus S$, because we require that the desired matching must not saturate a vertex from S. We let $\widetilde{\mathcal{Q}} = \widehat{\mathcal{Q}} \cap \mathcal{S}_X^{i-1}$. We now define a function $\mathsf{ovh} : \widetilde{\mathcal{Q}} \to \mathbb{N}$ (see Figure 6 for an intuitive illustration). For $Q_j \in \widetilde{\mathcal{Q}}$, we set $\mathsf{ovh}(Q_j) = |X_{j+1,i} \setminus S|$. To obtain $\mathcal{Q}$, we will delete those sets from $\widetilde{\mathcal{Q}}$ which will incur an "overhead" of crossings more than the "allowed" budget. Before constructing $\mathcal{Q}$, we first recall the following facts. By the definition of $\widetilde{\mathcal{Q}}$, we have $Q \in \mathcal{S}_X^{i-1}$. Moreover, from Observation 4 it follows that $S' \in \mathcal{S}_Y^{i-1}$ (as $y_i \notin S'$). We set $\mathcal{Q} = \{Q \in \widetilde{\mathcal{Q}} \mid \ell - \mathsf{ovh}(Q) \in \mathsf{Alw}_{i-1}(Q, S')\}$. Now we set $T[i, S, S', \ell]$ as follows.

$$T[i, S, S', \ell] = \begin{cases} 0, \text{ if } \mathcal{Q} = \emptyset, \\ \bigvee_{Q \in \mathcal{Q}} T[i-1, Q, S', \ell - \mathsf{ovh}(Q)], \text{ otherwise.} \end{cases}$$

▶ **Lemma 7.** *The computation of $T[i, S, S', \ell]$ in Case 2 is correct.*

Case 3: $x_i \notin S$ and $y_i \notin S'$. In this case, a matching, say M, which realizes $T[i, S, S', \ell]$, must saturate both the vertices x_i and y_i. Thus, M must have edges x_jy_i and $x_iy_{j'}$, where $j \leqslant i$ and $j' \leqslant i$. (Assuming x_i is adjacent to y_i in G, it can be the case that $j = j' = i$, in which case $x_iy_i \in M$.) We will thus have $T[i, S, S', \ell] = T_1[i, S, S', \ell] \vee T_2[i, S, S', \ell]$, where $T_1[i, S, S', \ell]$ and $T_2[i, S, S', \ell]$ are boolean variables that correspond respectively to the cases $j = j' = i$ and $j \neq i$ (and $j' \neq i$). We now define $T_1[i, S, S', \ell]$ and $T_2[i, S, S', \ell]$, formally.

Defining $T_1[i, S, S', \ell]$. Since $x_i \notin S$, we have $S \subseteq \widehat{X}_{i-1}$. As $y_i \notin S'$, we have $S' \subseteq \widehat{Y}_{i-1}$. By Observation 4, $S \in \mathcal{S}_X^{i-1}$ and $S' \in \mathcal{S}_Y^{i-1}$. Note that if a matching M that realizes $T[i, S, S', \ell]$ contains the edge x_iy_i (assuming x_iy_i is indeed an edge in the graph G), then $\mathtt{cr}(M) = \mathtt{cr}(M \setminus \{x_iy_i\})$. That is, no additional crossing is incurred by adding the edge x_iy_i to the matching $M \setminus \{x_iy_i\}$. Also, note that $\ell \in \mathsf{Alw}_{i-1}(S, S')$. With these observations, we define $T_1[i, S, S', \ell]$ as follows.

$$T_1[i, S, S', \ell] = \begin{cases} 0, \text{ if } x_iy_i \notin E(G), \\ T[i-1, S, S', \ell], \text{ otherwise.} \end{cases}$$

Defining $T_2[i, S, S', \ell]$. Now, to define $T_2[i, S, S', \ell]$, we proceed as in Case 2. For a vertex $x_j \in (N(y_i) \cap \widehat{X}_{i-1}) \setminus S$, let $Q_j = S \cup \{x_j\}$. We let $\widehat{\mathcal{Q}} = \{Q_j \mid x_j \in (N(y_i) \cap \widehat{X}_{i-1}) \setminus S\}$, and $\mathcal{Q} = \widehat{\mathcal{Q}} \cap \mathcal{S}_X^{i-1}$. Similarly, for a vertex $y_{j'} \in (N(x_i) \cap \widehat{Y}_{i-1}) \setminus S'$, let $R_{j'} = S' \cup \{y_{j'}\}$. We let $\widehat{\mathcal{R}} = \{R_{j'} \mid y_{j'} \in (N(x_i) \cap \widehat{Y}_{i-1}) \setminus S'\}$, and $\mathcal{R} = \widehat{\mathcal{R}} \cap \mathcal{S}_Y^{i-1}$. We will now construct a set of "crucial pairs" from $\mathcal{Q} \times \mathcal{R}$, for the computation of $T_2[i, S, S', \ell]$. Towards this, we define a function $\mathsf{ovh} : \mathcal{Q} \times \mathcal{R} \to \mathbb{N}$. We set $\mathsf{ovh}(Q_j, R_{j'}) = |X_{j+1,i} \setminus S| + |Y_{j'+1,i} \setminus S'| - 1$, for $Q_j \in \mathcal{Q}$ and $R_{j'} \in \mathcal{R}$. Finally, we let $\mathcal{C} = \{(Q, R) \in \mathcal{Q} \times \mathcal{R} \mid \ell - \mathsf{ovh}(Q, R) \in \mathsf{Alw}_{i-1}(Q, R)\}$. Now we set $T_2[i, S, S', \ell]$ as follows.

$$T_2[i, S, S', \ell] = \begin{cases} 0, \text{ if } \mathcal{C} = \emptyset, \\ \bigvee_{(Q,R) \in \mathcal{C}} T[i-1, Q, R, \ell - \mathsf{ovh}(Q, R)], \text{ otherwise.} \end{cases}$$

▶ **Lemma 8.** *The computation of $T[i, S, S', \ell]$ in Case 3 is correct.*

As observed earlier, (G, k) is a yes-instance of CM-PM if and only if there is $\ell \in [k]_0$, such that $T[n, \emptyset, \emptyset, \ell] = 1$. Note that for each $i \in [n]$, $S \in \mathcal{S}_X^i$, $S' \in \mathcal{S}_Y^i$, and $\ell \in \mathsf{Alw}_i(S, S')$, we can compute the entry $T[i, S, S', \ell]$ in time bounded by $n^{\mathcal{O}(1)}$. Moreover, the number of entries in our table is bounded by $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$ (see Lemma 3). Thus, the running time of the algorithm is bounded by $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$. The correctness of the algorithm follows from the correctness of base case and recursive formulae. Thus, we obtain the following theorem.

▶ **Theorem 9.** CM-PM *admits an algorithm running in time $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$.*

References

1. A. Karim Abu-Affash, Ahmad Biniaz, Paz Carmi, Anil Maheshwari, and Michiel H. M. Smid. Approximating the bottleneck plane perfect matching of a point set. *Comput. Geom.*, 48(9):718–731, 2015.
2. A. Karim Abu-Affash, Paz Carmi, Matthew J. Katz, and Yohai Trabelsi. Bottleneck non-crossing matching in the plane. *Comput. Geom.*, 47(3):447–457, 2014.
3. Jihad Al-Oudatallah, Fariz Abboud, Mazen Khoury, and Hassan Ibrahim. Overlapping Signal Separation Method Using Superresolution Technique Based on Experimental Echo Shape. *Advances in Acoustics and Vibration*, pages 1–9, 2017.
4. Victor Alvarez, Karl Bringmann, Radu Curticapean, and Saurabh Ray. Counting crossing-free structures. In *Symposuim on Computational Geometry 2012, SoCG '12, Chapel Hill, NC, USA, June 17-20, 2012*, pages 61–68, 2012.
5. Marc Benkert, Herman J. Haverkort, Moritz Kroll, and Martin Nöllenburg. Algorithms for Multi-criteria One-Sided Boundary Labeling. In *Graph Drawing, 15th International Symposium, GD 2007, Sydney, Australia, September 24-26, 2007. Revised Papers*, pages 243–254, 2007.
6. Therese C. Biedl, Franz-Josef Brandenburg, and Xiaotie Deng. Crossings and Permutations. In *Proceeding of the 13th International Symposium on Graph Drawing, GD*, volume 3843 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.

7 Édouard Bonnet, Tillmann Miltzow, and Paweł Rzążewski. Complexity of Token Swapping and Its Variants. *Algorithmica*, October 2017.

8 Sergio Cabello and Bojan Mohar. Adding One Edge to Planar Graphs Makes Crossing Number and 1-Planarity Hard. *SIAM J. Comput.*, 42(5):1803–1829, 2013.

9 John Gunnar Carlsson, Benjamin Armbruster, Saladi Rahul, and Haritha Bellam. A Bottleneck Matching Problem with Edge-Crossing Constraints. *Int. J. Comput. Geometry Appl.*, 25(4):245–262, 2015.

10 Xuanwu Chen and Ming S. Lee. A case study on multi-lane roundabouts under congestion: Comparing software capacity and delay estimates with field data. *Journal of Traffic and Transportation Engineering (English Edition)*, 3(2):154–165, 2016.

11 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

12 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

13 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry*, 4(5):235–282, 1994.

14 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

15 Vida Dujmovic, Michael R. Fellows, Michael T. Hallett, Matthew Kitching, Giuseppe Liotta, Catherine McCartin, Naomi Nishimura, Prabhakar Ragde, Frances A. Rosamond, Matthew Suderman, Sue Whitesides, and David R. Wood. On the Parameterized Complexity of Layered Graph Drawing. In *9th Annual European Symposium on Algorithms, ESA 2001, Proceedings*, pages 488–499, 2001.

16 Peter Eades and Sue Whitesides. Drawing graphs in two layers. *Theoretical Computer Science*, 131(2):361–374, 1994.

17 Peter Eades and Nicholas C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994.

18 Jack Edmonds. Paths, Trees and Flowers. *Canadian Journal of Mathematics*, pages 449–467, 1965.

19 Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical computer science*, 410(1):53–61, 2009.

20 Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.

21 Per Garder. Pedestrian safety at traffic signals: A study carried out with the help of a traffic conflicts technique. *Accident Analysis & Prevention*, 21(5):435–444, 1989.

22 M R Garey and D S Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W.H. Freeman, New York, 1979.

23 Michael R Garey and David S Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 4(3):312–316, 1983.

24 Martin Grohe. Computing crossing numbers in quadratic time. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 231–236, 2001.

25 Magnús M. Halldórsson, Christian Knauer, Andreas Spillner, and Takeshi Tokuyama. Fixed-Parameter Tractability for Non-Crossing Spanning Trees. In *Algorithms and Data Structures, 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007, Proceedings*, pages 410–421, 2007.

26 Godfrey H Hardy and Srinivasa Ramanujan. Asymptotic formulaæ in combinatory analysis. *Proceedings of the London Mathematical Society*, 2(1):75–115, 1918.

27 Petr Hlinený. Crossing number is hard for cubic graphs. *J. Comb. Theory, Ser. B*, 96(4):455–471, 2006.

28 Petr Hlinený and Marek Dernár. Crossing Number is Hard for Kernelization. In *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*, pages 42:1–42:10, 2016.

29 John E. Hopcroft and Robert Endre Tarjan. Efficient Algorithms for Graph Manipulation [H] (Algorithm 447). *Commun. ACM*, 16(6):372–378, 1973.

30 Klaus Jansen and Gerhard J. Woeginger. The Complexity of Detecting Crossingfree Configurations in the Plane. *BIT*, 33(4):580–595, 1993.

31 Michael Junger and Petra Mutzel. *Graph Drawing Software*. Springer-Verlag, Berlin, Heidelberg, 2003.

32 Ken-ichi Kawarabayashi and Bruce A. Reed. Computing crossing number in linear time. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 382–390, 2007.

33 Fabian Klute and Martin Nöllenburg. Minimizing Crossings in Constrained Two-Sided Circular Graph Layouts. In *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, pages 53:1–53:14, 2018.

34 Jan Kratochvíl, Anna Lubiw, and Jaroslav Nešetřil. Noncrossing Subgraphs in Topological Layouts. *SIAM J. Discret. Math.*, 4(2):223–244, March 1991.

35 Mukkai S Krishnamoorthy and Narsingh Deo. Node-deletion NP-complete problems. *SIAM Journal on Computing*, 8(4):619–625, 1979.

36 Martin I Krzywinski, Jacqueline E Schein, Inanc Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J Jones, and Marco A Marra. Circos: An information aesthetic for comparative genomics. *Genome Research*, 19(9):1639–1645, 2009.

37 J. Malik, J. Weber, Q. T. Luong, and D. Roller. Smart cars and smart roads. In *Proceedings 6th. British Machine Vision Conference*, pages 367–381, 1995.

38 Dániel Marx and Tillmann Miltzow. Peeling and Nibbling the Cactus: Subexponential-Time Algorithms for Counting Triangulations and Related Problems. In *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*, pages 52:1–52:16, 2016.

39 Tillmann Miltzow. Subset token swapping on a path and bipartite minimum crossing matchings. In *Order and Geometry Workshop, Problem booklet.*, pages 5–6, 2016. URL: `http://orderandgeometry2016.tcs.uj.edu.pl/docs/OG2016-ProblemBooklet.pdf`.

40 Tillmann Miltzow, Lothar Narins, Yoshio Okamoto, Günter Rote, Antonis Thomas, and Takeaki Uno. Approximation and Hardness of Token Swapping. In *24th Annual European Symposium on Algorithms, ESA 2016*, pages 66:1–66:15, 2016.

41 Monroe M. Newborn and William O. J. Moser. Optimal crossing-free Hamiltonian circuit drawings of K_n. *J. Comb. Theory, Ser. B*, 29(1):13–26, 1980.

42 Michael Osigbemeh, Michael Onuu, and Olumuyiwa Asaolu. Design and development of an improved traffic light control system using hybrid lighting system. *Journal of Traffic and Transportation Engineering (English Edition)*, 4(1):88–95, 2017. Special Issue: Driver Behavior, Highway Capacity and Transportation Resilience.

43 Marcus Schaefer. The Graph Crossing Number and its Variants: A Survey. *The Electronic Journal of Combinatorics*, 20, April 2013.

44 Carl Sechen. *VLSI placement and global routing using simulated annealing*, volume 54. Springer Science & Business Media, 2012.

45 Micha Sharir and Emo Welzl. On the Number of Crossing-Free Matchings, Cycles, and Partitions. *SIAM J. Comput.*, 36(3):695–720, 2006.

46 Paul Turán. A note of welcome. *Journal of Graph Theory*, 1(1):7–9, 1997.

47 Manuel Wettstein. Counting and enumerating crossing-free geometric graphs. *JoCG*, 8(1):47–77, 2017.

48 Katsuhisa Yamanaka, Erik D. Demaine, Takehiro Ito, Jun Kawahara, Masashi Kiyomi, Yoshio Okamoto, Toshiki Saitoh, Akira Suzuki, Kei Uchizawa, and Takeaki Uno. Swapping Labeled

Tokens on Graphs. In Alfredo Ferro, Fabrizio Luccio, and Peter Widmayer, editors, *Fun with Algorithms*, pages 364–375, 2014.

49 Katsuhisa Yamanaka, Erik D. Demaine, Takehiro Ito, Jun Kawahara, Masashi Kiyomi, Yoshio Okamoto, Toshiki Saitoh, Akira Suzuki, Kei Uchizawa, and Takeaki Uno. Swapping labeled tokens on graphs. *Theoretical Computer Science*, 586:81–94, 2015. Fun with Algorithms.

50 Lanbo Zheng and Christoph Buchheim. A New Exact Algorithm for the Two-Sided Crossing Minimization Problem. In *Proceedings of the First International Conference on Combinatorial Optimization and Applications, COCOA*, volume 4616 of *Lecture Notes in Computer Science*, pages 301–310. Springer, 2007.