Available online at www.sciencedirect.com

ScienceDirect

Journal of Discrete Algorithms 6 (2008) 313–323

JOURNAL OF DISCRETE ALGORITHMS

www.elsevier.com/locate/jda

Fixed parameter algorithms for ONE-SIDED CROSSING MINIMIZATION revisited ☆

Vida Dujmović [a,*], Henning Fernau [b], Michael Kaufmann [c]

[a] *Department of Mathematics and Statistics, McGill University, Montréal, Canada*
[b] *Universität Trier, FB IV, Abt. Informatik, 54286 Trier, Germany*
[c] *Universität Tübingen, WSI für Informatik, Sand 13, 72076 Tübingen, Germany*

Received 5 May 2005; received in revised form 4 December 2006; accepted 14 December 2006

Available online 24 May 2007

Abstract

We exhibit a small problem kernel for the ONE-SIDED CROSSING MINIMIZATION problem. This problem plays an important role in graph drawing algorithms based on the Sugiyama layering approach. Moreover, we improve on the search tree algorithm developed in [V. Dujmović, S. Whitesides, An efficient fixed parameter tractable algorithm for 1-sided crossing minimization, Algorithmica 40 (2004) 15–31] and derive an $\mathcal{O}(1.4656^k + kn^2)$ algorithm for this problem, where k upperbounds the number of tolerated edge crossings in the drawings of an n-vertex graph.

Keywords: Graph drawing; Crossing minimization; 2-layer drawings; $\mathcal{FPT}$; Parameterized algorithms

1. Introduction and problem definition

A graph $G = (V, E)$ with vertex set V and edge set $E \subseteq V \times V$ is *bipartite* if there is a partition of V into two sets V_1 and V_2 such that $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, and $E \subseteq V_1 \times V_2$. Here, we study the k-ONE-SIDED CROSSING MINIMIZATION problem, k-OSCM for short, that is defined as follows.

Given: A simple n-vertex bipartite graph $G = (V_1, V_2, E)$ and a linear order $<_1$ on V_1.
Parameter: A nonnegative integer k.
Question: Is there a linear order $<$ on V_2 such that, when the vertices of V_1 are placed on a line (also called *layer*) in the order induced by $<_1$ and the vertices of V_2 are placed on a second layer (parallel to the first one) in the order induced by $<$, then drawing a straight-line segment for each edge in E will introduce no more than k (pairwise) edge crossings?

☆ An extended abstract covering parts of this paper has appeared in the Proceedings of the 11th International Symposium on Graph Drawing (GD 2003), see [V. Dujmović, H. Fernau, M. Kaufmann, Fixed parameter algorithms for one-sided crossing minimization revisited, in: Proc. of International Symposium on Graph Drawing (GD'03), Lecture Notes Comput. Sci., vol. 2912, Springer, 2004, pp. 332–344].

* Corresponding author.
E-mail addresses: vida@cs.mcgill.ca (V. Dujmović), fernau@informatik.uni-trier.de (H. Fernau), mk@informatik.uni-tuebingen.de (M. Kaufmann).

1570-8667/$ – see front matter
doi:10.1016/j.jda.2006.12.008

We denote by OSCM the optimization version of this problem, that is, the problem that asks for a linear order $<$ on V_2 that induces the minimum possible number of edge crossings.

1.1. *Improving on graph drawing algorithms*

(k-)OSCM is the key procedure in the Sugiyama algorithm [12], which is the well-known layout framework for drawings graphs on layers. After the first phase (the assignment of the vertices to layers), the order of the vertices within each layer has to be fixed such that the number of the corresponding crossings of the edges with endpoints in two adjacent layers is minimized. Finally, the concrete position of the vertices within each layer is determined according the computed order. The crossing minimization step, although the most essential in the Sugiyama approach, is an $\mathcal{NP}$-complete problem. The most commonly used crossing minimization method is the layer-by-layer sweep heuristics where, starting from $i = 1$, the order for layer L_i is fixed and an order for L_{i+1} that minimizes the number of crossings amongst the edges with endpoints in L_i and L_{i+1} is determined. After increasing index i to the maximum layer index, this process is repeated from the back with decreasing indices. In each step, an OSCM problem has to be solved. Unfortunately, this seemingly elementary problem is $\mathcal{NP}$-complete [8], even for sparse graphs [9].

This fundamental graph drawing problem attracted several researchers from the area of fixed-parameter tractable ($\mathcal{FPT}$) algorithms [3]. The first approaches to the more general variant of this problem have been published by Dujmović et al. in [4] and [5]. The last one has been greatly improved by Dujmović and Whitesides [7] who achieved an $\mathcal{O}(1.6182^k n^2)$ algorithm for k-OSCM using search tree techniques. There has been a similar race to get better approximation algorithms for OSCM. To our knowledge, the best one has been reported by Nagamochi [10] with an approximation factor of 1.4664.

In this paper, we derive an $\mathcal{O}(1.4656^k + kn^2)$ algorithm for k-OSCM. The exponential part of the running time, resulting from the search tree part of our algorithm, is significantly lower than that of [7]. Moreover, we exhibit a small problem kernel for this problem, which has not been done before. In particular, with the aid of some reduction rules, we can arrive at an instance of k-OSCM that has at most $3k^2$ vertices.

The remainder of the paper is organized as follows. After fixing terminology in Section 2, we derive a problem kernel for k-OSCM in Section 3. In Section 4, we present our algorithm for k-OSCM. The correctness and the running time analysis of the algorithm is based on the study presented in Sections 5 and 6. Concluding remarks are found in Section 7.

2. Preliminaries

We start with some formalities. Consider a graph $G = (V, E)$. For each vertex $v \in V$, let $N(v)$ denote the set of vertices adjacent to v and let $\deg(v) := |N(v)|$ denote the degree of v in G. For $U \subseteq V$, let $N(U) = \bigcup_{u \in U} N(u)$. The subgraph of G induced by a set of vertices $U \subseteq V$ is denoted by $G[U]$.

We will call a bipartite graph $G = (V_1, V_2, E)$ together with linear orders on V_1 and on V_2 a *drawing* of G. This formulation implicitly assumes a drawing where the vertices of V_1 and V_2 are represented by distinct points on two horizontal lines (layers), the line corresponding to V_1 being above the line corresponding to V_2, with vertices in each set appearing from left to right according to their respective linear orders. The edges of G are represented by straight-line segments between their endpoints. A *crossing* in a drawing is given by a pair of edges that intersect at some point other than a possible common endpoint. Note that two edges (x, a) and (y, b), with $x, y \in V_1$ and $a, b \in V_2$, cross in a drawing if and only if $x < y$ and $b < a$. If $a < b$ for two vertices in V_1 or in V_2, then a is *to the left* of b, and b is *to the right* of a. A linear order on V_2 that minimizes the number of crossings subject to the fixed linear order on V_1 is called an *optimal ordering* and the corresponding drawing of G is called an *optimal drawing*. Since isolated vertices play no part in edge crossings, in what follows we disregard isolated vertices of the input graph G.

If $|V_2| = 2$, then there are only two different drawings. This suggests the useful notion of a *crossing number* for a pair of vertices in V_2. Given a bipartite graph $G = (V_1, V_2, E)$ with $|V_2| > 1$, for any pair of vertices $a, b \in V_2$, define c_{ab} to be the number of crossings in the drawing of $G[\{a, b\} \cup N(a) \cup N(b)]$ when $a < b$ is assumed. We say that $\{a, b\}$ forms a c_{ab}/c_{ba} *pattern*. If $c_{ab} \leqslant c_{ba}$, then $a < b$ is called the *cheaper ordering*. Dujmović and Whitesides [7] have shown that, in any optimal ordering $<$ of the vertices of V_2, each pair $\{a, b\}$ that forms a $0/j$ pattern where $j > 0$, is ordered as $a < b$. This means that all the pairs that form $0/j$ patterns appear in their cheaper ordering in any optimal drawing. The example in Fig. 1 demonstrates some of these notions.

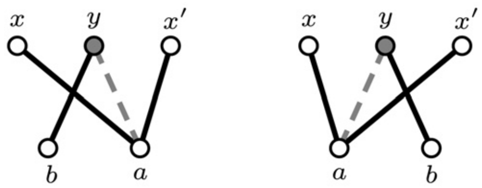

Fig. 1. Let $G = (V_1, V_2, E)$ with $V_1 = (x < y < x')$, $V_2 = \{a, b\}$, and $E = \{\{a, x\}, \{a, x'\}, \{b, y\}\}$. This figure illustrates two drawings corresponding to $b < a$ and $a < b$ orderings, and demonstrates that in this case $c_{ab} = c_{ba} = 1$. Thus, $\{a, b\}$ forms a $1/1$ pattern. The dashed line is optional.

A solution to a k-OSCM instance is a linear order on V_2 with at most k crossings in the associated drawing, if such a linear order exists. Our algorithm builds such a linear order, by picking a pair of vertices $\{a, b\}$ of V_2 and ordering them as $a < b$ or $b < a$, based on some rules, and reducing k appropriately. Thus at each stage of our algorithm, we have a poset $P = (V_2, <)$. We now recall some notation related to partial orders.

A *partial order* is an irreflexive, asymmetric and transitive relation. A partially ordered set (or *poset*), denoted by $P = (V, <)$, is a set of elements V taken together with a partial order $<$ on it. For two distinct elements $a, b \in V$, if either $a < b$ or $b < a$ in P, then the pair $\{a, b\}$ is *comparable* in P; else the pair is *incomparable*. A partial order is *linear* if every pair of elements of V is comparable.

Definition 1. In a poset $P = (V, <)$, a pair of incomparable elements $\{a, b\} \in V$ is

- *dependent with respect to c* if at least one of the pairs $\{a, c\}$ or $\{b, c\}$ is incomparable in P;
- *dependent* (*in P*) if there is $c \in V \setminus \{a, b\}$ such that $\{a, b\}$ is dependent with respect to c;
- *transitive with respect to c* if either pair $\{a, c\}$ or pair $\{b, c\}$ is comparable but not both;
- *transitive* (*in P*) if there is $c \in V \setminus \{a, b\}$ such that $\{a, b\}$ is transitive with respect to c.

For the remainder of this paper, we will consider an *annotated version* of k-OSCM, where the instance I comprises of $G = (V_1, V_2, E)$, a linear order $(V_1, <_1)$ and a partial order $(V_2, <)$. The task is then to find a linear order $(V_2, <')$ that extends $<$ and that introduces no more than k pairwise edge crossings. We will also sometimes allow negative parameter values to indicate a failure case. The *size* of the parameterized instance $\langle I, k\rangle$ of the annotated version will be measured by $\text{size}(\langle I, k\rangle) = |V_1| + |V_2| + |E| + |V_2 \times V_2 \setminus <| + k$.

3. Getting a small kernel

Our goal in this section is to reduce a given problem instance of k-OSCM to an equivalent problem instance (called *kernel*) that has size bounded by a function on the parameter k only. We achieve that by developing a set of reduction rules. Applying a reduction rule to a parameterized instance $\langle I, k\rangle$ results in a parameterized instance $\langle I', k'\rangle$ with $\text{size}(\langle I', k'\rangle) < \text{size}(\langle I, k\rangle)$. A reduction rule is *valid* if I has a solution of size at most k if and only if I' has a solution of size at most k'. If R is a valid reduction rule, then a parameterized instance $\langle I, k\rangle$ is called *R-reduced* if and only if R is not applicable to that instance.

When settling the ordering between a and b, we also say that we are *committing* $a < b$ or $b < a$. In what follows, whenever we commit a pair of vertices, say $a < b$, we reduce k not only by c_{ab}, but also by c_{cd} for each pair $\{c, d\}$ that gets committed to $c < d$ due to transitivity. If this process results in $k < 0$, we replace the problem instance with the empty graph with parameter $k = -1$ to indicate failure. We refer to these operations as *parameter accounting*.

Dujmović and Whitesides [7] have shown that in any optimal drawing, the pairs that form $0/j$ patterns appear in their cheaper ordering. This justifies the following reduction rule.

Reduction Rule `RR1`: For each pair of vertices $\{a, b\} \subseteq V_2$ that forms a $0/j$ pattern with $j > 0$, commit $a < b$.

Consider now a set $S \subseteq V_2$ such that, for all vertices v in S, $N(v) = N(S)$. It is simple to observe that arbitrarily permuting the vertices of S in any drawing cannot affect the total number of crossings. This observation leads to the following reduction rule.

Reduction Rule `RR2`: For each pair of vertices $\{a, b\} \in V_2$ with $N(a) = N(b)$, (arbitrarily) commit $a < b$, and do the parameter accounting.

Notice that if $c_{ab} = c_{ba} = 0$, then (disregarding isolated vertices) $\deg(a) = \deg(b) = 1$ and $N(a) = N(b)$. Therefore, in any `RR1`- and `RR2`-reduced instance of k-OSCM, a pair of vertices $\{a, b\}$ is incomparable in the poset $P = (V_2, <)$ only if both $c_{ab} > 0$ and $c_{ba} > 0$.

Reduction Rule `RRlarge`: If $c_{ab} > k$, then commit $b < a$ and do the parameter accounting.

If $c_{ba} \leqslant k$, the rule is clearly valid. If $c_{ba} > k$, then the parameter accounting results in the parameter becoming negative integer and thus a trivially unsolvable instance is returned. Thus the rule is valid. Moreover, if a vertex $a \in V_2$ has $\deg(a) \geqslant 2k + 1$, then for each vertex $b \in V_2$, $c_{ab} > k$ or $c_{ba} > k$.

Reduction Rule `RRLO1`: If a vertex $v \in V_2$ is comparable in $P = (V_2, <)$ with each vertex of $V_2 \setminus v$, then remove v from V_2, and let $P = (V_2 \setminus v, <)$.

This rule is clearly valid, since v is comparable with all the vertices of V_2 and as such plays no further part in the cost of an optimal ordering. After having exhaustively applied `RRLO1`, there is, for each vertex $a \in V_2$, another vertex $b \in V_2$ such that the pair $\{a, b\}$ is incomparable in P. By the positivity assumption, committing $a < b$ or $b < a$ will reduce k by at least one. Hence, in a `RR1`-, `RR2`-, and `RRLO1`-reduced instance of k-OSCM, $|V_2| \leqslant 2k$.

The following rule further reduces the size of V_2. More importantly, as will become clear later, `RRLO2` plays a rôle in reducing the base of the exponent in the running time of our search tree algorithm.

Reduction Rule `RRLO2`: If $\{a, b\}$ is an incomparable pair that is not dependent in $P = (V_2, <)$ with $c_{ab} \leqslant c_{ba}$, then commit $a < b$, and do the parameter accounting.

For any $c \in V_2 \setminus \{a, b\}$, either $\alpha < c$ or $c < \alpha$ is known for $\alpha \in \{a, b\}$, since $\{a, b\}$ is not dependent. Thus either $a < c$ and $b < c$, or $c < a$ and $c < b$. Hence, in every linear order that extends $<$, a and b will be immediate neighbors. Thus, the ordering of a and b can only affect crossings of edges incident to a and b; therefore, in any minimal linear order $<'$ extending $<$, we will have $a <' b$ whenever $c_{ab} \leqslant c_{ba}$, breaking ties arbitrarily. Thus rule `RRLO2` is valid.

Our reduction rules suggest the following kernelization algorithm.

Algorithm 1 *(Kernelization for k-OSCM).*
`Compute the crossing numbers` c_{ab} `and` c_{ba} `for all pairs` $\{a, b\}$.
`Apply reduction rules RR1, RR2 exhaustively.`
`Apply reduction rules RRLO1, RRLO2, RRlarge exhaustively.`

Theorem 1. *Given an instance of the k-OSCM problem, Algorithm 1 computes its problem kernel $G = (V_1, V_2, E)$ with $|V_1| \leqslant 3k^2$, $|V_2| \leqslant \frac{3}{2}k$, and $|E| \leqslant 3k^2$ in $\mathcal{O}(kn^2)$ time.*

Proof. Applying `RR1` and `RR2` to k-OSCM results in a poset $P = (V_2, <)$ where each incomparable pair of vertices has both crossing numbers positive. Thus there are at most k incomparable pairs of vertices in P. After exhaustive application of rules `RRLO1` and `RRLO2`, each vertex in V_2 is dependent in P, and thus $|V_2| \leqslant \frac{3}{2}k$. If a vertex $a \in V_2$ has $\deg(a) \geqslant 2k + 1$, then for each vertex $b \in V_2$, $c_{ab} > k$ or $c_{ba} > k$. Therefore, after exhaustive application of `RRlarge` in combination with `RRLO1`, each vertex in V_2 has degree at most $2k$. Thus $|E| \leqslant 2k|V_2| \leqslant 3k^2$ and $|V_1| \leqslant 2k|V_2| \leqslant 3k^2$.

Since computing the crossing numbers is the predominant computational part (taking time $\mathcal{O}(kn^2)$ [7]), the kernelization algorithm runs in $\mathcal{O}(kn^2)$ time. □

Notice that, strictly speaking, we are getting an *annotated kernel* by our reduction rules, see [1] for a further discussion on variants of the notion of a kernel in parameterized complexity. However, since we will continue with a search tree algorithm that deals with the annotated version of k-OSCM, this is the appropriate notion of kernelization.

4. The search tree algorithm

As is the standard practice when developing search tree based $\mathcal{FPT}$ algorithms, each node of a search tree is associated with a problem instance of the annotated version of k-OSCM. all the pairs of vertices of V_2 committed thus far, and the parameter k' gives the remaining number of allowed edge crossings, that is, $k' = k - \sum_{v<u} c_{vu}$. Before a recursive call of the search tree algorithm itself, committing one case (out of possibly two cases) involves updating P and doing the parameter accounting. After having processed the branches, NO will be returned if and only if all branches yielded NO.

Algorithm 2 *(A parameterized algorithm for k-OSCM).*

```
// PREPROCESSING:
– kernelize (Algorithm 1)
– apply reduction rule RR3 exhaustively;

// SEARCH TREE: (Recursive calls start at step 0.)
FOREACH node of the search tree with annotated instance
    (G = (V1, V2, E), <1, P = (V2, <), k') DO
0: Apply RRLO1, RRLO2, RRlarge exhaustively.
1: IF k' < 0, THEN return NO.
2: IF in P there is an incomparable i/j pattern {a, b} with i + j ⩾ 4,
     THEN branch on two cases: a < b and b < a;
3: ELSE IF in P there is a dependent 2/1 pattern {a, b},
     THEN branch on two cases: a < b and b < a;
4: ELSE IF there is a 1/1 pattern {a, b} in P
     THEN commit a < b and recurse;
5: ELSE return YES.
```

The remainder of this paper is dedicated to proving the correctness and the claimed running time of Algorithm 2. Before diving into details, we now give a brief overview for each issue.

Correctness. The correctness of the algorithm is easily verified assuming the correctness of reduction rule RR3 and step 4 of the algorithm. Consider a pair of incomparable vertices $\{a, b\}$ in step 4 of the algorithm. By the kernelization and steps 2 and 3 of the algorithm, $\{a, b\}$ forms either a $2/1$ pattern that is not dependent, or a $1/1$ pattern. Since RRLO2 commits all the pairs that are not dependent, $\{a, b\}$ forms a $1/1$ pattern. Thus, all remaining incomparable pairs in step 4 form $1/1$ patterns. Therefore, no matter how these pairs are ordered, the resulting ordering has the same cost. To complete the proof of correctness of Algorithm 2 we need to verify that the rule RR3 is valid. That will follow from the analysis given in Section 5.3.

Running time. The running time of the preprocessing is dominated by the kernelization. Thus, by Theorem 1, the preprocessing runs in $\mathcal{O}(kn^2)$ time. However, the running time of $\mathcal{FPT}$ algorithms is dominated by the part exponential in parameter k. In our case, that part is bounded by the number of nodes in the search tree. Denote that number by $s(k)$. Then the running time of Algorithm 2 is $\mathcal{O}(s(k) + kn^2)$—Niedermeier and Rossmanith's analysis [11] of the related rekernelization method implies that this is true despite the fact that more than a constant amount of work is done in each node of the search tree. Each internal node of our search tree has two branches. If one branch lowers the parameter k' by b_1, and the other by b_2, we denote the corresponding *branching vector* by (b_1, b_2). We will prove that each internal node of our search tree obeys $b_1 + b_2 \geqslant 4$ and $b_1, b_2 > 0$, which in turn will allow us to prove that $s(k) < 1.4656^k$—we do that next. This is in contrast to the search tree with $b_1 + b_2 \geqslant 3$ and $b_1, b_2 > 0$, as derived in [7], that gives $s(k) < 1.6181^k$.

Suppose that we know that in each node of our search tree, $b_1 + b_2 \geqslant 4$, and $b_1, b_2 > 0$. Then in the worse case, each node has a branching vector $(2, 2)$ or $(3, 1)$. The recurrence corresponding to the $(2, 2)$ branching vector is $s(k) = 2s(k-2) + \mathcal{O}(1)$. Solving this recurrence gives $s(k) < 1.4143^k$. The recurrence corresponding to the $(3, 1)$ branching vector is $s(k) = s(k-3) + s(k-1) + \mathcal{O}(1)$. Solving this recurrence gives $s(k) < 1.4656^k$. Thus in the worst case, $s(k) \leqslant 1.4656^k$. Therefore, to complete the running time analysis we need to prove that in each node of our search tree, $b_1 + b_2 \geqslant 4$, and $b_1, b_2 > 0$.

Clearly, $b_1 + b_2 \geqslant 4$ for all the internal nodes of the search tree created by step 2 of the algorithm. If all the $2/1$ patterns could be committed deterministically, then step 3 would never be executed and it would immediately follow that $b_1 + b_2 \geqslant 4$ for all the nodes of the search tree. Unfortunately, not all 2/1 patterns can be committed deterministically, as will become clear from the analysis in Section 5.3. However, we show in Section 5.3 that some $2/1$ patterns can be committed deterministically, giving rise to rule RR3. For those remaining $2/1$ patterns for which step 3 does get executed, we prove in the main lemma (Lemma 4) that each such pair is transitive. Thus, an additional pair gets committed due to transitivity, allowing us to conclude that $b_1 + b_2 \geqslant 4$ in step 3, as well. Having rule RR3 is instrumental in the proof of the main lemma, and thus in bounding the size of the search tree.

In order to derive and prove the correctness of rule RR3, as well as prove that $b_1 + b_2 \geqslant 4$, we need to analyze $1/1$ and $2/1$ patterns. This is exhibited in the next section.

5. 1/1 and 2/1 patterns

5.1. Structural properties of 1/1 *and* 2/1 *patterns*

In the following illustrations, as a convention, we will label vertices from the first layer by letters x, y and vertices from the second layer by letters a, b. Furthermore, we will draw neighbors of a as non-filled circles and neighbors of b as filled-in circles, allowing also overlays as in Fig. 3.

Lemma 1. *Suppose an instance of k-OSCM has a pair $\{a, b\}$ that forms a $1/1$ or a $2/1$ pattern and has $c_{ba} = 1$. Then such a pair must be a part of the subgraph as depicted in Fig. 2, where each remaining neighbor of a (if any) must be to the right of y (or y itself), while each remaining neighbor of b (if any) must be to the left of x (or x itself). (Otherwise, $c_{ba} > 1$.)*

For the relatively simple combinatorial proofs of the following two lemmas, refer to [6].

Lemma 2. *Suppose an instance of k-OSCM has a pair $\{a, b\}$ that forms a $1/1$ pattern, that is $c_{ab} = c_{ba} = 1$. Then such a pair must be part of one of two distinct subgraphs. (The two basic subgraphs depicted in Figs. 3 and 4 can be obtained by enhancing the situation sketched in Fig. 2.)*

(1) *a and b are each adjacent to x and y only. In other words,* $\deg(a) = \deg(b) = 2$ *and* $N(a) = N(b)$*, as illustrated in Fig. 3.*
(2) *Two sub-cases arise:* (a) *If* $\deg(b) = 1$*, then a has (besides x) another neighbor x' to the right of y. In addition to x and x', a may only be adjacent to y.* (b) *If* $\deg(a) = 1$*, then b has (besides y) another neighbor y' to the left of x. In addition to y and y', b may only be adjacent to x. Both situations are illustrated in Fig. 4.*

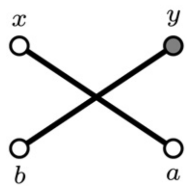

Fig. 2. An elementary crossing.

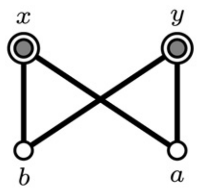

Fig. 3. A simple enforced crossing.

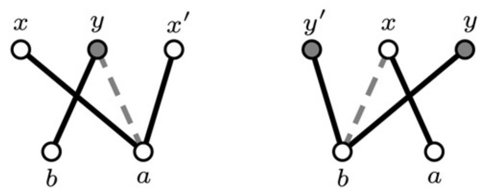

Fig. 4. A second case for a $1/1$ pattern; the dashed lines are optional.

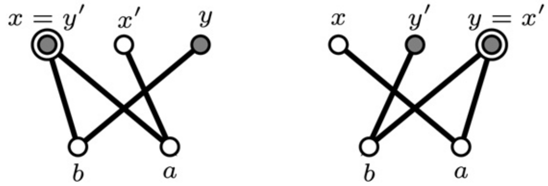

Fig. 5. $2/1$ patterns, case 1.

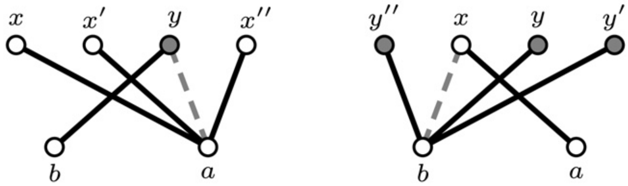

Fig. 6. $2/1$ patterns, case 2.

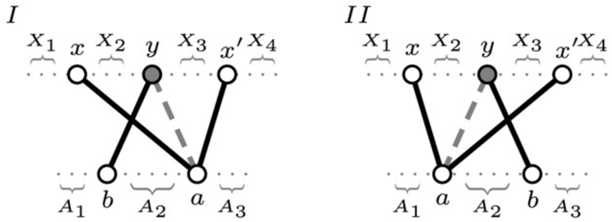

Fig. 7. Schematic analysis for a $1/1$ pattern.

Lemma 3. *Suppose an instance of k-OSCM has a pair $\{b, a\}$ that forms a $2/1$ pattern, that is $c_{ab} = 1$ and $c_{ba} = 2$. Then such a pair must be a part of one of the two distinct subgraphs as described in the two cases below.*

(1) *In the first case, illustrated in Fig. 5, neither a nor b can have any other neighbors.*
(2) *In the second case, illustrated in the left part of Fig. 6, in addition to x, x' and x'', vertex a may only be adjacent to y, while b has no other neighbors. Hence, a has degree three or four and b has degree one. The second case, illustrated in the right part of Fig. 6, can be symmetrically interpreted.*

5.2. The tabular analysis technique

The reduction rule RR2 presented in Section 3 commits the $1/1$ patterns characterized in the first case of Lemma 2. Unfortunately, the second type of $1/1$ patterns characterized in Lemma 2 does not admit a similar simple resolution.

The following analysis of $1/1$ patterns is included for pedagogical reasons mainly: it presents the simplest example of a schematic *tabular analysis* and is therefore one step on the way of a systematic analysis of all possible situations. Admittedly, in this case the analysis fails in the sense that it does not show that one of the two possible ways to commit $1/1$ pattern is always better. Nonetheless, a lesson learned here is that in order to improve on the earlier search tree algorithm [7], we have to take special care of this case.

We now demonstrate the schematic tabular analysis on the $1/1$ pattern $\{a, b\}$ as characterized in the second case of Lemma 2. The following refers to Fig. 7. Let I denote any drawing where $b < a$ and let II denote the drawing obtained from drawing I by swapping the ordering of a and b. Let the total number of crossings in I and II be denoted by c_I and c_{II}, respectively. We partition the remaining vertices on the first layer into vertex sets X_1 through X_4. Similarly, we partition the remaining vertices on the second layer into vertex sets A_1, A_2, A_3.

For each of the two drawings we create the tables T_I and T_{II} (see Fig. 8). (The tables correspond to drawings I and II in Fig. 7 without the dotted edges.) These tables read as follows: if there are $m_{i,j}$ edges connecting vertices from X_i with vertices from A_j then there are $T_x[i, j] \cdot m_{i,j}$ crossings between these $m_{i,j}$ edges and the edges shown

T_I	A_1	A_2	A_3
X_1	0	$\boxed{1}$	3
X_2	1	$\boxed{2}$	2
X_3	2	$\boxed{1}$	1
X_4	3	$\boxed{2}$	0

T_{II}	A_1	A_2	A_3
X_1	0	$\boxed{2}$	3
X_2	1	$\boxed{1}$	2
X_3	2	$\boxed{2}$	1
X_4	3	$\boxed{1}$	0

Fig. 8. The tabular analysis for Fig. 7.

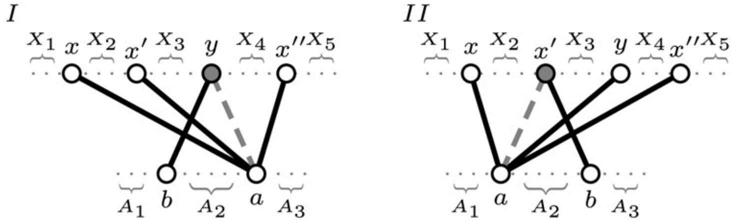

Fig. 9. Schematics for the second 2/1 pattern.

in the above sketches for case $x \in \{I, II\}$. It is clear that the columns labeled A_1 and A_3 are identical in both tables: swapping a and b can only affect the relative order of vertices in A_2 compared to a and b. The entries that differ in the two tables are framed by boxes.

Tables like these can help decide when a pair of vertices can be deterministically committed, and here is how. Let the total number of crossings in drawing I and II be denoted by c_I and c_{II}, respectively. Then,

$$c_I - c_{II} = c_{ba} - c_{ab} + \sum_{i,j} m_{i,j}\big(T_I[i,j] - T_{II}[i,j]\big)$$

and similarly,

$$c_{II} - c_I = c_{ab} - c_{ba} + \sum_{i,j} m_{i,j}\big(T_{II}[i,j] - T_I[i,j]\big).$$

Note that the terms in the sum are non-zero only for the framed numbers in the two tables. If we can show that $c_I - c_{II} > 0$, then we could conclude that any optimal drawing shows $a < b$ and we could commit $\{a, b\}$ deterministically (similarly if we can show $c_{II} - c_I > 0$). Unfortunately, such conclusion is not possible in case of the above 1/1 patterns. That is because $c_I - c_{II} = -m_{1,2} + m_{2,2} - m_{3,2} + m_{4,2}$ and $c_{II} - c_I = m_{1,2} - m_{2,2} + m_{3,2} - m_{4,2}$, and thus, whether $c_I - c_{II} > 0$ or $c_I - c_{II} > 0$ may depend on the values of $m_{i,j}$s. Thus we cannot give any unconditional deterministic choice rule in this situation. Similar problems arise when the "optional" dotted edges between a and y are present.

5.3. *Analyzing* 2/1 *patterns*

Consider first the 2/1 patterns $\{a, b\}$ characterized in the second case of Lemma 3. That case is similar to (but more complicated than) the second pattern of Lemma 2 and hence may not admit a deterministic solution. This is confirmed by the schematic tabular analysis depicted in Figs. 9 and 10. The tables T_I and T_{II} in Fig. 10 correspond to drawings I and II without the dotted edges in Fig. 9.

The situation is more favorable for 2/1 patterns $\{a, b\}$ characterized in the first case of Lemma 3. Consider first the sub-case where a has a neighbor distinct from x and y, that is, refer to the left drawing in Fig. 5. Again, let I denote any drawing where $b < a$ and let II denote the drawing obtained from drawing I by swapping a and b (see Fig. 11). Let the total number of crossings in I and II be denoted by c_I and c_{II}, respectively. Figs. 11 and 12 correspond to this case, that is, the first case of Lemma 3.

T_I	A_1	A_2	A_3
X_1	0	$\boxed{1}$	4
X_2	1	2	3
X_3	2	$\boxed{3}$	2
X_4	3	2	1
X_5	4	$\boxed{3}$	0

T_{II}	A_1	A_2	A_3
X_1	0	$\boxed{3}$	4
X_2	1	2	3
X_3	2	$\boxed{1}$	2
X_4	3	2	1
X_5	4	$\boxed{1}$	0

Fig. 10. Tabular analysis for the situation from Fig. 9.

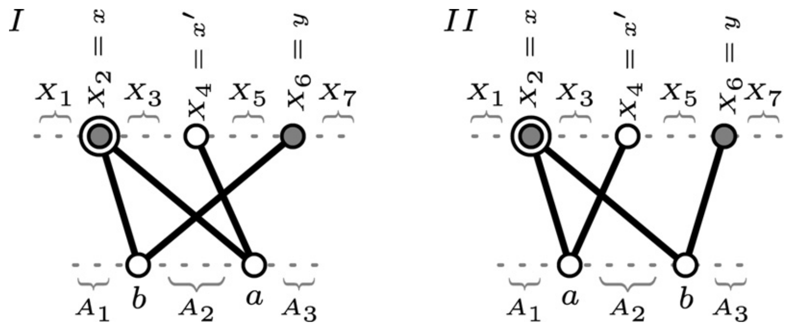

Fig. 11. Schematics for the first 2/1 pattern.

T_I	A_1	A_2	A_3
X_1	0	2	4
X_2	0	1	2
X_3	2	2	2
X_4	2	$\boxed{2}$	1
X_5	3	$\boxed{3}$	1
X_6	3	$\boxed{2}$	0
X_7	4	2	0

T_{II}	A_1	A_2	A_3
X_1	0	2	4
X_2	0	1	2
X_3	2	2	2
X_4	2	$\boxed{1}$	1
X_5	3	$\boxed{1}$	1
X_6	3	$\boxed{1}$	0
X_7	4	2	0

Fig. 12. Tabular analysis for the situation from Fig. 11.

It follows that $c_I - c_{II} = c_{ba} - c_{ab} + m_{4,2} + 2m_{5,2} + m_{6,2} > 0$, since $m_{4,2} + 2m_{5,2} + m_{6,2} \geqslant 0$ and $c_{ba} - c_{ab} = 1$. This implies that whenever this situation described in Lemma 3 arises, any optimal drawing shows $a < b$.

The first case in Lemma 3 where b has a neighbor distinct from x and y is symmetric; the same analysis applies. This justifies the following reduction rule.

Reduction Rule `RR3`: For each pair of vertices $\{a, b\} \in V_2$ that forms a $2/1$ pattern as characterized in the first case of Lemma 3, commit $a < b$, and do the parameter accounting.

6. Putting it all together

The analysis presented in the previous section proves that rule `RR3` is valid. Therefore, by the arguments presented in Section 4, Algorithm 2 is correct.

By the running time arguments presented in Section 4, to complete the running time analysis for Algorithm 2, we need to show that the branching vector (b_1, b_2) in each internal node of the search tree has $b_1 + b_2 \geqslant 4$ and $b_1, b_2 > 0$. We do that in the next two lemmas.

Lemma 4 *(Main Lemma). Let $\{a, b\}$ be a pair of dependent vertices that forms a 2/1 pattern in step 3 of Algorithm 2. Then $\{a, b\}$ is a transitive pair.*

Proof. Since $\{a, b\}$ is dependent in P, there must be a vertex c such that $\{a, c\}$ or $\{b, c\}$ are incomparable in P. It suffices to show that one of these two pairs are comparable in P. In step 3 of the algorithm, the only remaining incomparable pairs are 1/1 patterns of the second type in Lemma 2 and 2/1 patterns of the second type in Lemma 3. Therefore, let without loss of generality $\deg(a) = 3$ (or $\deg(a) = 4$) and $\deg(b) = 1$. If $\deg(c) \geqslant 2$, then[1] $c_{ac} + c_{ca} \geqslant 4$ and $\{a, c\}$ is comparable either by RR1 or by step 2 of the algorithm. Therefore, $\deg(c) = 1$. In that case, the pair $\{b, c\}$ forms a $0/i$ pattern and its ordering is settled by either RR1 or RR2. □

Lemma 5. *The branching vector (b_1, b_2) in each internal node of the search tree associated with Algorithm 2 has $b_1 + b_2 \geqslant 4$ and $b_1, b_2 > 0$.*

Proof. Based on the reduction rules RR1 and RR2, in each node of a search tree all the incomparable (and dependent) pairs form i/j patterns such that $i > 0$ and $j > 0$. Thus in each node $b_1, b_2 > 0$. Furthermore, all the nodes branched in step 2, have $i + j \geqslant 4$ and thus have $b_1 + b_2 \geqslant 4$. The remaining nodes are branched in step 3. Each such node branches on a dependent 2/1 pattern $\{a, b\}$. By the previous lemma, committing either $a < b$ or $b < a$ determines, without loss of generality, the ordering of a pair $\{a, c\}$. Having been incomparable at step 3 initially, the pair $\{a, c\}$ has $c_{ac}, c_{ca} \geqslant 1$. Therefore, we have $b_1 + b_2 \geqslant 4$ in this case, too. □

Finally we can conclude:

Theorem 2. *Algorithm 2 solves the k-OSCM problem in $\mathcal{O}(1.4656^k + kn^2)$ time.*

7. Conclusions

In this paper, we present a new search tree based parameterized algorithm for the k-OSCM problem that runs in $\mathcal{O}(1.4656^k + kn^2)$ time. It remains to be determined whether further progress is possible, especially in further lowering the base of the exponent in the running time. Our present case analysis shows at two places a branching behavior which matches the above time complexity bound, namely when branching at 3/1 and at 2/1 patterns. A detailed analysis of 3/1 patterns may be worthwhile, but would probably be very tedious, requiring considerations of all possible configurations.

Acknowledgements

The authors are grateful for hints of F. Rosamond, P. Shaw and M. Suderman on draft versions of this paper. Thanks to an anonymous reviewer whose comments greatly improved the readability of the paper.

References

[1] F. Abu-Khzam, H. Fernau, Kernels: annotated, proper and induced, in: H.L. Bodlaender, M. Langston (Eds.), International Workshop on Parameterized and Exact Computation IWPEC, in: Lecture Notes Comput. Sci., vol. 4169, Springer, 2006, pp. 264–275.
[2] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis, Graph Drawing: Algorithms for the Visualization of Graphs, Prentice-Hall, 1999.
[3] R. Downey, M. Fellows, Parameterized Complexity, Springer, 1999.
[4] V. Dujmović, M. Fellows, M. Hallet, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. Rosamond, M. Suderman, S. Whitesides, D. Wood, A fixed-parameter approach to 2-layer planarization, Algorithmica 45 (2006) 159–182.

[1] That is because $c_{vw} + c_{wv} = \deg(v)\deg(w) - |N(v) \cap N(w)|$ for all $v, w \in V_2$ (see [2, Chapter 9]). This implies that if $\deg(w) \leqslant \deg(v)$, then $(\deg(v) - 1)\deg(w) \leqslant c_{vw} + c_{wv} \leqslant \deg(v)\deg(w)$.

[5] V. Dujmović, M. Fellows, M. Hallet, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. Rosamond, M. Suderman, S. Whitesides, D. Wood, On the parameterized complexity of layered graph drawing, in: Proc. European Symposium on Algorithms (ESA'01), in: Lecture Notes Comput. Sci., vol. 2161, Springer, 2001, pp. 488–499.
[6] V. Dujmović, H. Fernau, M. Kaufmann, Fixed parameter algorithms for one-sided crossing minimization revisited, in: Proc. of International Symposium on Graph Drawing (GD '03), in: Lecture Notes Comput. Sci., vol. 2912, Springer, 2004, pp. 332–344.
[7] V. Dujmović, S. Whitesides, An efficient fixed parameter tractable algorithm for 1-sided crossing minimization, Algorithmica 40 (2004) 15–31.
[8] P. Eades, N.C. Wormald, Edge crossings in drawings of bipartite graphs, Algorithmica 11 (1994) 379–403.
[9] X. Muñoz, W. Unger, I. Vrt'o, One sided crossing minimization is $\mathcal{NP}$-hard for sparse graphs, in: Proc. of International Symposium on Graph Drawing (GD'01), in: Lecture Notes Comput. Sci., vol. 2265, Springer, 2006, pp. 115–123.
[10] H. Nagamochi, An improved bound on the one-sided minimum crossing number in two-layered drawings, Discrete and Computational Geometry 33 (2005) 569–591.
[11] R. Niedermeier, P. Rossmanith, A general method to speed up fixed-parameter-tractable algorithms, Information Processing Letters 73 (2000) 125–129.
[12] K. Sugiyama, S. Tagawa, M. Toda, Methods for visual understanding of hierarchical system structures, IEEE Transactions on Systems, Man and Cybernetics 11 (1981) 109–125.