Width Notions for Ordering-Related Problems

Emmanuel Arrighi
University of Bergen, Norway
emmanuel.arrighi@uib.no

Henning Fernau
University of Trier, Germany
fernau@uni-trier.de

Mateus de Oliveira Oliveira
University of Bergen, Norway
mateus.oliveira@uib.no

Petra Wolf
University of Trier, Germany
wolfp@informatik.uni-trier.de

Abstract

We are studying a weighted version of a linear extension problem, given some finite partial order ρ, called Completion of an Ordering. While this problem is NP-complete, we show that it lies in FPT when parameterized by the interval width of ρ. This ordering problem can be used to model several ordering problems stemming from diverse application areas, such as graph drawing, computational social choice, or computer memory management. Each application yields a special ρ. We also relate the interval width of ρ to parameterizations such as *maximum range* that have been introduced earlier in these applications, sometimes improving on parameterized algorithms that have been developed for these parameterizations before. This approach also gives some practical sub-exponential time algorithms for ordering problems.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Dynamic programming; Mathematics of computing → Combinatorial optimization

Keywords and phrases Parameterized algorithms, interval width, linear extension, one-sided crossing minimization, Kemeny rank aggregation, grouping by swapping

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2020.9

Funding *Emmanuel Arrighi*: Research Council of Norway (Grant no. 274526), IS-DAAD (Grant no. 309319).
Henning Fernau: DAAD PPP (Grant no. 57525246).
Mateus de Oliveira Oliveira: Trond Mohn Foundation, Research Council of Norway (Grant no. 288761), IS-DAAD (Grant no. 309319).
Petra Wolf: DFG project FE 560/9-1, DAAD PPP (Grant no. 57525246).

1 Introduction

Many computational problems can be phrased as the task of arranging a collection of combinatorial objects into a minimum-cost linear order that satisfies certain constraints. Examples of natural problems that fall in this category are One-Sided Crossing Minimization (OSCM), a prominent problem in the field of graph drawing and VLSI design [4, 32, 45, 50, 52], Grouping by Swapping (GbS), a problem with applications in computer memory management [15, 28, 55], and Kemeny Rank Aggregation (KRA), a prominent problem in the field of computational social choice [19, 36]. A natural parameter that arises when studying problems such as OSCM, GbS and KRA from the perspective of parameterized complexity theory is the cost k of a solution. In particular, the best algorithm

for OSCM, parameterized by the cost of a solution k, is the algorithm due to Kobayashi and Tamaki [38] which runs in time[1] $\mathcal{O}^*(2^{\sqrt{2k}})$ and the best single-exponential algorithm for KRA runs in time $\mathcal{O}^*(1.403^k)$ [51], while sub-exponential algorithms of type $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k})})$ have been proposed in [35], with some unclear constant hidden in the $\mathcal{O}$-notation of the exponent. Not surprisingly, they have been devised with substantially distinct sets of techniques.

In this paper, significantly extending the ideas started out in [23, 25], we leverage the Completion of an Ordering problem (CO) to provide a unified framework for the study of several cost-parameterized ordering problems. In this problem, we are given a partial order ρ on a set V, and a function $\mathfrak{c} : V \times V \to \mathbb{N}$ assigning costs to incomparable pairs, and the goal is to compute a minimum-cost linear extension of ρ. Interestingly, a natural *structural* parameter that arises in this context is the pathwidth of the cocomparability graph of the input partial order ρ. This graph has V as vertex-set and there is an undirected edge between vertices v and v' if and only if v and v' are not related in the partial order. Our main result states that CO, parameterized by the interval width w of the input partial order, can be solved in time $\mathcal{O}^*(2^{\mathrm{w}})$. Additionally, our algorithm is optimal under ETH. Using our main result, together with reductions from OSCM, GbS and KRA to PCO, the natural restriction of CO to positive costs, we obtain algorithms for these three problems (parameterized by width, or by the standard parameter, or by other problem-specific structural parameters) whose running times often match or improve on the best algorithms for the three problems.

When reducing OSCM or GbS to PCO, the partial order one obtains is an interval order, meaning that the cocomparability graph of this order is an interval graph. Interval orders play an important role in partial order theory due to the fact that their interval width can be computed in linear time. Additionally, they find applications in many contexts of practical relevance such as scheduling, online and packing algorithms, see [54]. Inspired by this, we define the Positive Completion of an Interval Ordering (PCIO) problem, a version of PCO where the input partial order is required to be an interval order. In this restricted version, our main algorithm for CO parameterized by interval width can be converted into a sub-exponential $\mathcal{O}^*(2^{\sqrt{2k}})$-time FPT algorithm for PCIO, parameterized by cost k.

Our width-based approach also allows us to improve on a parameterized algorithm for KRA based on the parameter *maximum range* (of a candidate) as introduced and discussed in [5]. Further, it can be used to show that GbS is also fixed parameter tractable when parameterized by a parameter called scope coincidence degree, a natural parameter in the context of strings. This gives the first algorithmic use of this structural string parameter.

Our approach for CO is built on dynamic programming on a path decomposition of the cocomparability graph of the partial order. Notice that this path decomposition structure has been recently exploited for counting the number of linear extensions by Eiben *et al.* [22]. Here, we use this approach to find the cheapest linear extension.

2 Preliminaries

In this section, we collect the basic notions of this paper. $\mathbb{N}$ denotes the set of non-negative integers and $\mathbb{N}_{>0}$ denotes the set of positive integers. Given $r \in \mathbb{N}_{>0}$, we write $[r] \doteq \{1, \ldots, r\}$.

Notation on Partial Orders. Let V be a set. A *partial order* over V is a reflexive, anti-symmetric and transitive binary relation $\rho \subseteq V \times V$. We say that ρ is a *linear order* if additionally, for each $(x, y) \in V \times V$, either $(x, y) \in \rho$ or $(y, x) \in \rho$. A *strict partial order*

[1] Recall that the $\mathcal{O}^*$-notation suppresses polynomial factors.

over V is an irreflexive and transitive binary relation $\sigma \subseteq V \times V$. By adding the identity relation I_V, $\rho \doteq \sigma \cup I_V$ becomes a partial order, and conversely from a partial order ρ on V, we can define $\sigma \doteq \rho \setminus I_V$ as a strict partial order. Hence, we will occasionally use the term linear order also for the corresponding strict order, often denoted as $<_\rho$ for reasons of clarity. Notice that for finite base sets V, we can specify a linear order $<_f$ by a bijection $f : [|V|] \to V$, with the understanding that $f(i) <_f f(j)$ if and only if $i < j$, i.e., if the number i is smaller than the number j. Such a bijection f is also called a *ranking* in the following. Conversely, any linear order τ on Σ defines a bijection $f_\tau : [|V|] \to V$.

Given two partial orders $\rho, \tau \subseteq V \times V$, τ is an *extension* of ρ if $\rho \subseteq \tau$. If τ is also a linear order on V, then τ is a *linear extension* of ρ. Given a linear order τ on V, let $\min_\tau(V)$ be the minimum element in V with respect to τ and $\max_\tau(V)$ be the maximum element in V with respect to τ. Given a subset $T \subseteq V$ and a partial order $\rho \subseteq V \times V$, let $\rho|_T \doteq \rho \cap T \times T$ be the restriction of ρ to T. A linear order $\tau \subseteq T \times T$ is a *linear extension of ρ on T* if τ is a linear extension of $\rho|_T$. We define $\mathsf{Lin}(\rho, T)$ to be the set of linear extensions of ρ on T.

Notation on Graphs. Given an undirected graph $G = (V, E)$ and a vertex $v \in V$, we let $N(v) \doteq \{u \mid u \in V, (v, u) \in E\}$ be the neighborhood of v.

A path decomposition of a graph $G = (V, E)$ is a sequence $D = (B_1, B_2, \ldots, B_r)$ of subsets of V, such that the following conditions are satisfied.

- $\bigcup_{1 \leq i \leq r} B_i = V$.
- For each edge $(u, v) \in E$, there is an $i \in [r]$ such that $u, v \in B_i$.
- For each $i, j, k \in [r]$ with $i < j < k$, $B_i \bigcap B_k \subseteq B_j$.

The *width* of D is defined as $\mathrm{w}(D) = \max_{i \in [r]} |B_i| - 1$. The *pathwidth*, $pw(G)$, of G is the minimum width of a path decomposition of G.

Partial Orders and Interval Width. Given a (strict) partial order $\rho \subseteq V \times V$, the undirected graph $G_\rho \doteq (V, E)$ with $E \doteq \{\{u, v\} \in V \times V \mid u \neq v, (u, v) \notin \rho, (v, u) \notin \rho\}$ is the *cocomparability graph* of ρ. An *interval order* is a strict partial order $\iota \subseteq V \times V$ whose elements $v \in V$ are represented by half-open intervals $I_v = [l_v, r_v)$ on the real line with $(u, v) \in \iota \iff r_u \leq l_v$. $\{I_v \mid v \in V\}$ is called an *interval representation* of ι. The cocomparability graph G_ι is the intersection graph of $\{I_v \mid v \in V\}$ and is hence an interval graph. It is known [29] that interval graphs are exactly the cocomparability graphs that do not contain an induced cycle of length four. The *interval width* of a partial order $\rho \subseteq V \times V$ is defined as $\mathrm{iw}(\rho) \doteq \min\{\mathrm{w}(\iota) | \iota \text{ interval order}, \iota \subseteq \rho\}$, where $\mathrm{w}(\iota)$ is the maximum size of an antichain of ι. By Theorem 2.1 from [31], $pw(G_\rho) = \mathrm{iw}(\rho) - 1$. Conversely, for any graph $G = (V, E)$, $pw(G) = \min\{\omega(H) \mid H \text{ is an interval graph}, V(H) = V, E(H) \supseteq E\} - 1$, where $\omega(H)$ is the size of the largest clique in H. For more information on interval orders, we refer to textbooks and survey articles such as [26, 54].

3 Completion of an Ordering (CO)

Below, we formally define the Completion of an Ordering problem, generalizing Positive Completion of an Ordering (PCO) introduced in [16, Sec. 8] and [23, Sec. 6.4].

Problem name: Completion of an Ordering (CO)
Given: A partial order $\rho \subseteq V \times V$, a cost function $\mathfrak{c} : V \times V \to \mathbb{N}$, and $k \in \mathbb{N}$.
Output: Is there a linear order $\tau \supseteq \rho$ with $\mathfrak{c}(\tau \setminus \rho) = \sum_{(x,y) \in \tau \setminus \rho} \mathfrak{c}(x, y) \leq k$?

In the PCO problem, the cost function needs to satisfy the following condition: for all pairs $(x, y) \in V \times V$ such that x and y are incomparable in ρ, $\mathfrak{c}(x, y) > 0$.

Let us shortly discuss the *cost parameter* k: By the result of Dujmovic, Fernau and Kaufmann [16] (for details, see [23]), PCO can be solved in time $\mathcal{O}^*(1.52^k)$ and admits a linear-size kernel. The best known algorithm for PCO, whose running time is $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k}\log(k))})$, was obtained in [25] by relating PCO to the Feedback Arc Set Problem in Tournaments, or FAST for short, that allows for subexponential algorithms due to [1]. Here, we are presenting an algorithm for a variation of this problem that runs in time $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k})})$ and is relatively straightforward to implement. We also present a branching algorithm that runs in time $\mathcal{O}^*(1.42^k)$, improving on [23]. Our algorithms are based on the interval width of ρ.

3.1 CO, parameterized by pathwidth

Let $G = (V, E)$ be a graph, $\rho \subseteq V \times V$ be a (strict) partial order on the vertices of G and $D = (B_1, \ldots, B_r)$ be a path decomposition of G. We call D *consistent* with ρ if there is no pair of vertices $(x, y) \in \rho$ with $\max\{i \in [r] \mid y \in B_i\} < \min\{i \in [r] \mid x \in B_i\}$. Thus, if x is smaller than y in ρ, then y cannot be forgotten in D before x is introduced in D. The *consistent pathwidth*, $cpw(G, \rho)$, of G is the minimum width of a path decomposition of G consistent with ρ. We will be interested in particular in the consistent pathwidth $cpw(G_\rho, \rho)$.

▶ **Theorem 1.** *Given a partial order ρ over a set V, a cost function $\mathfrak{c} : V \times V \to \mathbb{N}$ and a width-*w *path decomposition D of the cocomparability graph G_ρ that is consistent with ρ, one can solve an instance $(\rho, \mathfrak{c}, k)$ of the CO problem in time $\mathcal{O}(|V| \cdot \mathrm{w} \cdot 2^{\mathrm{w}} \cdot \log(k) + |V|^2 \cdot \log(k))$.*

The remainder of this subsection is dedicated to the proof of Theorem 1.

Let us explain why our pathwidth measure can be seen as a *distance to triviality* parameterization in the context of CO. A trivial instance of CO is a linear order, as it has cost zero. Then, the cocomparability graph is an independent set and has consistent pathwidth 0[2]. In the opposite case, if the input partial order is empty, then the cocomparability graph is a clique and has consistent pathwidth $|V| - 1$. It is also worth noticing that it is NP-hard to determine the pathwidth of a cocomparability graph, together with an optimal path decomposition, as observed in [31].

Notation on Path Decompositions. Let $D = (B_1, B_2, \ldots, B_r)$ be a path decomposition of a graph G. We say that $[r]$ is the set of positions of D and that r is the length of D. For each position i, we say that B_i is the i-th bag of D. For each $i \in [r]$, $i > 1$, we say that B_i is an *introduce bag* if $B_i = B_{i-1} \cup \{v\}$ and that B_i is a *forget bag* if $B_i = B_{i-1} \setminus \{v\}$. We say that the path decomposition $D = (B_1, B_2, \ldots, B_r)$ is *nice* if for each $i \in [r]$, B_i is either an introduce bag or a forget bag and $|B_1| = 1$ and $B_r = \emptyset$. It can be shown that, given any path decomposition $D = (B_1, B_2, \ldots, B_r)$ of width w of a graph G, one can construct in time $\mathcal{O}(r \cdot \mathrm{w}(D))$ a nice path decomposition of G of width at most w. In a nice path decomposition, for every vertex of V, there is a bag that introduces it and a bag that forgets it, so the length of a nice path decomposition is $2 \cdot |V|$. For each position $i \in [r]$, we let $L_i = \bigcup_{1 \leq j \leq i-1} B_j \setminus B_i$ be the set of vertices that have been forgotten (lost) up to position i.

▶ **Lemma 2.** *Let ι be an interval order over V and $\{I_v \mid v \in V\}$ be an interval representation of ι. One can derive a minimum width path decomposition of G_ι consistent with ι from $\{I_v \mid v \in V\}$ of width $\mathrm{w}(\iota) - 1$ in time $\mathcal{O}(\mathrm{w}(\iota) \cdot |V|)$.*

[2] In Lemma 5, we show that consistent pathwidth is equal to pathwidth.

Proof. For each element v in V, we let l_v and r_v be the left and right endpoints of I_v. For every point x on the real line $\mathbb{R}$ that corresponds to an endpoint of one or more intervals, we associate a bag $B_x = \{v \mid x \in I_v\}$. Then, we order the bags following the order of l_v and r_v on the real line. For each element $v \in V$, $v \in B_{l_v}$. Given three bags B_x, B_y, B_z such that $x \leq y \leq z$, we have that $B_x \cap B_z = \{v \mid x \in I_v\} \cap \{v \mid z \in I_v\} = \{v \mid l_v \leq x \leq z < r_v\} \subseteq \{v \mid l_v \leq y < r_v\} = B_y$. For each edge $(u, v) \in E(G_\iota)$, I_u and I_v intersect, therefore, we have either $l_v \in I_u$ or $l_v \in I_u$. If $l_v \in I_u$ then $u, v \in B_{l_v}$, similarly if $l_u \in I_v$ then $u, v \in B_{l_u}$. So this construction builds a path decomposition. We call this path decomposition D. Now, we will show that D is consistent with ι. More precisely, we will show that for each pair $(u, v) \in \iota$, $\max\{x \in \mathbb{R} \mid u \in B_x\} < \min\{x \in \mathbb{R} \mid v \in B_x\}$. For each $(u, v) \in \iota$, we have $l_u < r_u \leq l_v$, $\max\{x \in \mathbb{R} \mid u \in B_x\} < r_u \leq l_v \leq \min\{x \in \mathbb{R} \mid v \in B_x\}$. Therefore, D is consistent with ι. Note that each bag is a clique, therefore, this is a path decomposition of minimum width. A clique in G_ι is an antichain of ι and each antichain of ι forms a clique in G_ι. Therefore, we have that D has width $\mathrm{w}(\iota) - 1$. ◀

We will refer to this decomposition as the path decomposition *derived* from the interval order ι.

▶ **Lemma 3.** *Let $G = (V, E)$ be a graph. Given a partial order ρ on V and a path decomposition D of G of width* w *and length r that is consistent with ρ, one can construct in time $\mathcal{O}(\mathrm{w} \cdot r)$ a nice path decomposition of width* w *that is consistent with ρ.*

Given a path decomposition D, one can get a nice path decomposition by introducing before each bag B several new bags that will forget one by one each vertex forgotten by B and introduce each new vertex in B one by one. If D is consistent with ρ, then the new path decomposition is also consistent with ρ. For the cocomparability graph, we can further show:

▶ **Lemma 4.** *Let ρ be a partial order on a set V, G_ρ be the cocomparability graph of ρ and D be a path decomposition of G_ρ consistent with ρ, then D is consistent with any extension of ρ.*

▶ **Lemma 5.** *Let $G_\rho = (V, E)$ be the cocomparability graph of a partial order $\rho \subseteq V \times V$. Then $pw(G_\rho) = cpw(G_\rho, \rho)$.*

Proof. By definition we have $pw(G_\rho) \leq cpw(G_\rho, \rho)$. We will show that $cpw(G_\rho, \rho) \leq \mathrm{iw}(\rho) - 1$ and use the fact that $pw(G_\rho) = \mathrm{iw}(\rho) - 1$ (Theorem 2.1 from [31]). By definition of $\mathrm{iw}(\rho)$, we can find an interval order ι such that $\mathrm{iw}(\rho) = \mathrm{w}(\iota)$ and $\iota \subseteq \rho$. Let $\{I_v \mid v \in V\}$ be an interval representation of ι. Then G_ι is the intersection graph of $\{I_v \mid v \in V\}$. Then by Lemma 2, the path decomposition D of G_ι derived from $\{I_v \mid v \in V\}$ is consistent with ι and has width $\mathrm{w}(\iota) - 1$. From Lemma 4, we know that D is also consistent with the extension ρ of ι. We conclude $cpw(G_\rho, \rho) \leq cpw(G_\iota, \iota) = \mathrm{w}(\iota) - 1 = \mathrm{iw}(\rho) - 1 = pw(G_\rho)$. ◀

Dynamic Programming Algorithm. Let $\rho \subseteq V \times V$ be a partial order over a set V, $\mathfrak{c} : V \times V \to \mathbb{N}$ be a cost function and S and T be two subsets of V such that for each pair $(s, t) \in S \times T$, $(t, s) \notin \rho$. We define $\mathfrak{c}(S, T) = \sum_{(s,t) \in (S \times T) \setminus \rho} \mathfrak{c}(s, t)$, this is the cost of having elements of S before elements of T. For every linear extension τ of ρ on T, we let $\mathfrak{c}(\tau) = \sum_{(a,b) \in \tau \setminus \rho|_T} \mathfrak{c}(a, b)$ be the cost of τ. We define $\mathrm{opt}(T) = \min\{\mathfrak{c}(\tau) \mid \tau \in \mathsf{Lin}(\rho, T)\}$. Our goal is to find $\mathrm{opt}(V)$.

Let D be a path decomposition of width w of the graph G_ρ consistent with ρ. By Lemma 3, we can assume without loss of generality that D is nice.

For each position $1 \leq i \leq 2 \cdot |V|$ in the path decomposition, we compute and store $\mathfrak{c}(L_i, \{v\})$ for every vertex $v \in B_i$ such that for each $u \in L_i$ $(v,u) \notin \rho$ in table $T_i^{\mathfrak{c}}$ and $\mathrm{opt}(L_i \cup T)$ for each $T \subseteq B_i$ in table T_i^{opt}. For every vertex $v \in B_i$ such that for each $u \in L_i$ $(v,u) \notin \rho$, $\mathfrak{c}(L_i, \{v\})$ is the cost of having v after the vertices forgotten at position i if this is compatible with ρ and for each $T \subseteq B_i$, $\mathrm{opt}(L_i \cup T)$ is the minimum cost of a linear extension on $L_i \cup T$. We have $L_{2\cdot|V|} \cup B_{2\cdot|V|} = V$. So, to find the solution, it is enough to inductively construct these two tables. The induction basis is trivial: $L_1 = \emptyset$ and $|B_1| = 1$, so that $\mathfrak{c}(L_1, \{v\}) = 0$ for every vertex $v \in B_1$ in table $T_1^{\mathfrak{c}}$ and $\mathrm{opt}(L_i \cup T) = 0$ for both $T = \emptyset$ and $T = B_1$ in table T_1^{opt}. The following two lemmas explain the induction step of the algorithm.

▶ **Lemma 6.** *Let $i \in [2, \ldots, 2 \cdot |V|]$. Given a table $T_{i-1}^{\mathfrak{c}}$ that lists the values of $\mathfrak{c}(L_{i-1}, \{v\})$ for every $v \in B_{i-1}$, one can compute $\mathfrak{c}(L_i, \{v\})$ for every $v \in B_i$ in time $\mathrm{w} \cdot \log(k)$ in order to build the table $T_i^{\mathfrak{c}}$.*

▶ **Lemma 7.** *Let $i \in [2, \ldots, 2 \cdot |V|]$. Given a table $T_i^{\mathfrak{c}}$ that lists the values of $\mathfrak{c}(L_i, \{v\})$ for every $v \in B_i$ such that for each $u \in L_i$ $(v,u) \notin \rho$ and a table T_{i-1}^{opt} that lists the values of $\mathrm{opt}(L_{i-1} \cup T)$ for every $T \subseteq B_{i-1}$, one can compute in $\mathcal{O}(\mathrm{w} \cdot 2^{\mathrm{w}} \cdot \log(k))$ time the value of $\mathrm{opt}(L_i \cup T)$ for all $T \subseteq B_i$ in order to build the table T_i^{opt}.*

Proof. The cost can be arbitrarily large, therefore, the addition of two costs is done in time $\mathcal{O}(\log(k))$. First, we compute $\mathfrak{c}(T, \{v'\})$ for $v' \in B_i$ and for $T \subseteq B_i \setminus \{v'\}$, and store the values in an auxiliary table T^{aux}. This computation can be done in $\mathcal{O}(\mathrm{w} \cdot 2^{\mathrm{w}} \cdot \log(k))$ time. Now there are two cases:

- If B_i forgets a vertex v, then $L_i = L_{i-1} \cup \{v\}$; for each subset $T \subseteq B_i$, $\mathrm{opt}(L_i \cup T) = \mathrm{opt}(L_{i-1} \cup T \cup \{v\})$ and this value is in the table T_{i-1}^{opt}, as $T \cup \{v\} \subseteq B_{i-1}$.
- If B_i introduces a vertex v, then $L_i = L_{i-1}$ and $B_i = B_{i-1} \cup \{v\}$. Given a subset T of B_i, if $v \notin T$, then $\mathrm{opt}(L_i \cup T)$ is already in the table T_{i-1}^{opt}. Suppose $v \in T$. For all $u \in L_i$, there is no edge between u and v in G_ρ, and as D is consistent with ρ, we have $(u,v) \in \rho$. So in any linear extension of ρ on $L_i \cup T$, the maximum element is a maximal element of T (with respect to ρ). Then we have, by testing all possible maximum elements v':

$$\begin{aligned}\mathrm{opt}(L_i \cup T) &= \min_{v' \in \max_\rho(T)} \{\mathrm{opt}(L_i \cup T \setminus \{v'\}) + \mathfrak{c}(L_i \cup T \setminus \{v'\}, \{v'\})\} \\ &= \min_{v' \in \max_\rho(T)} \{\mathrm{opt}(L_i \cup T \setminus \{v'\}) + \mathfrak{c}(L_i, \{v'\}) + \mathfrak{c}(T \setminus \{v'\}, \{v'\})\}\end{aligned}$$

  where $\max_\rho(T) = \{v \in T \mid \forall u \in T, (v,u) \notin \rho\}$ is the set of maximal elements of T with respect to ρ. The second and third terms are in the tables $T_i^{\mathfrak{c}}$ and T^{aux}, respectively. If $v' = v$, then the first term can be looked up in table T_{i-1}^{opt}. By walking through all $T \subseteq B_i$ with increasing cardinality (recall that always $v \in T$), we can inductively compute $\mathrm{opt}(L_i \cup T)$, as this provides the first term. As inductive basis, consider $T = \{v\}$, in which case $\mathrm{opt}(L_i \cup T) = \mathrm{opt}(L_i \cup \{v\}) = \mathrm{opt}(L_i) + \mathfrak{c}(L_i, \{v\})$. The first term is already in the table T_{i-1}^{opt}. The computation of T_i^{opt} can be done in time $\mathcal{O}(\mathrm{w} \cdot 2^{\mathrm{w}} \cdot \log(k))$.

This explains how to build the table T_i^{opt}. ◀

Since $L_{2\cdot|V|} \cup B_{2\cdot|V|} = V$, the dynamic programming algorithm can provide an optimal solution and runs in time $\mathcal{O}(|V| \cdot \mathrm{w} \cdot 2^{\mathrm{w}} \cdot \log(k))$. The size of the cost function given as input is $|V|^2 \cdot \log(k)$. Reading the cost function gives the second part of the running time. This proves Theorem 1.

3.2 Further Algorithmic Consequences

The relation between variants of FAST and CO range in both directions. One direction (solving PCO with the help of FAST) was exploited in [25]. We are now explaining a reverse reduction. The CONSTRAINED FAST problem [9, 57] is defined as follows: The arc set of a given tournament graph is split into fixed arcs A_{fix} and free arcs A_{free}. The task is to remove at most k free arcs such that the resulting graph becomes acyclic. We know that every arc in A_{free} that contradicts the transitivity of A_{fix} needs to be removed. Therefore, we assume that A_{fix} gives a transitive relation on the set of vertices and that it is acyclic, so that it defines a partial order ρ on the vertex set V. By defining the following cost function, we can solve CONSTRAINED FAST with any CO algorithm: For arcs $(x, y) \in A_{\text{free}}$, we set $\mathfrak{c}(x, y) = 0$. For arcs (x, y) such that $(y, x) \in A_{\text{free}}$, we set $\mathfrak{c}(x, y) = 1$. By the tournament condition, for each edge $\{x, y\}$ of G_ρ, $\mathfrak{c}(x, y) \in \{0, 1\}$ and $\mathfrak{c}(y, x) \in \{0, 1\}$ are defined, with $\mathfrak{c}(x, y) + \mathfrak{c}(y, x) = 1$. As FAST is a well-known NP-complete problem, this also shows NP-completeness for CO (even with costs $0, 1$ only) and similarly, we obtain NP-completeness for PCO, even with costs from the set $[2] = \{1, 2\}$.

Completing an interval ordering is easier. Consider the following restriction of PCO:

Problem name: POSITIVE COMPLETION OF AN INTERVAL ORDERING (PCIO)
Given: An interval order $\iota \subseteq V \times V$ over a set V, a cost function $\mathfrak{c} : V \times V \to \mathbb{N}$ satisfying $\forall x, y \in V : ((x, y) \notin \iota \wedge (y, x) \notin \iota) \implies \mathfrak{c}(x, y) > 0$, and an integer $k \in \mathbb{N}$.
Output: Is there a linear order $\tau \supseteq \iota$ with $\mathfrak{c}(\tau \setminus \iota) \leq k$?

This variation has two more restrictions compared to CO: the cost between two incomparable elements must not be zero and the partial order is an interval order. These two restrictions allow us to get better bounds for our dynamic programming algorithm.

▶ **Theorem 8.** *An instance $(\iota, \mathfrak{c}, k)$ of PCIO is solvable in time $\mathcal{O}(k \cdot 2^{\sqrt{2k}} \cdot \log(k) + |V|^2 \cdot \log(k))$.*

The following is an outline of our algorithm, called DP-PCIO.

1. Construct G_ι, if G_ι has more than k edges then stop with "NO". This can be done in time $|V|^2$. This is justified, because $\mathfrak{c}(x, y) > 0$ for each incomparable pair $\{x, y\}$.
2. Construct a nice path decomposition D consistent with ι. If the width of D is more than $\sqrt{2k}$, then stop with "NO", as a large clique was detected.
3. Compute $\text{opt}(V)$ by a dynamic programming algorithm based on the path decomposition D. If the current optimum solution is bigger than k, then stop with "NO". If the computation is successful and $\text{opt}(V) \leq k$ then answer "YES". Otherwise answer "NO".

We will prove several lemmas to show Theorem 8. To apply our dynamic programming algorithm, we need consistency.

▶ **Lemma 9.** *Given an interval order ι, one can construct in linear time a path decomposition of G_ι consistent with ι of minimum width.*

Let $D = (B_1, \ldots, B_{2|V|})$ be the nice path decomposition consistent with ι we got by applying Lemma 3 on the path decomposition of Lemma 9. Clearly, each bag in D is a clique.

▶ **Lemma 10.** *Assume that G_ι has at most k edges. Let $H = \lceil \sqrt{2k} \rceil + 1$ and, for $2 \leq h \leq H$, let $c_h \doteq |\{i : |B_i| = h\}|$. Then we have $c_h \leq k/(h-1) - h/2 + 1$.*

Finally, we show how our considerations also help to improve the running time of a simple branching algorithm. The algorithm works as follows: it picks an edge in the cocomparability graph and considers orienting it both ways. As long as there are *profitable edges* that cause at least a cost of two in each branch, we keep on branching. Costs can also be implicitly caused, as we modify the partial order ρ and hence transitivity must be maintained. We can use Theorem 8 when there are no more profitable edges because of the following lemma.

▶ **Lemma 11.** *After exhaustively branching at all profitable edges, G_ρ is an interval graph.*

This is the key to the following improvement on the branching algorithm described in [23]. Notice that in practice, branching algorithms tend to be faster at least for small parameter values, due to the smaller constants in the basis of the (sub-)exponential functions that upper-bound the running times.

▶ **Theorem 12.** *PCO can be solved in time $\mathcal{O}^*(\sqrt{2}^k)$ by a branching algorithm.*

4 One-Sided Crossing Minimization (OSCM)

Given a bipartite graph G with bipartition (V_1, V_2), a two-layer drawing of G is a drawing such that vertices of V_1 and V_2 are placed on two parallel lines and edges are represented as straight lines between the vertices. A two-layer drawing can be specified by two linear orders τ_1 of V_1 and τ_2 of V_2. A crossing in a two-layer drawing is a pair of edges that intersect each other in a point that is not a vertex. The number of crossings is defined by the order of V_1 and V_2 on the lines. The One-Sided Crossing Minimization problem consists in placing vertices of one part V_2 of the bipartite graph, given an ordering of the other part V_1, that minimizes the number of crossings. This problem is a key sub-problem for drawing hierarchical graphs [3, 4, 32, 45] or producing row-based VLSI layouts [50, 52].

Problem name: One-Sided Crossing Minimization (OSCM)
Given: A bipartite graph $G = (V_1, V_2, E)$, a linear order τ_1 on V_1 and $k \in \mathbb{N}$
Output: Is there a linear order τ_2 on V_2 such that, in the two-layer drawing specified by (τ_1, τ_2), at most k edge crossings incur?

The problem is known to be NP-complete [21] even in sparse graphs [44] and FPT in the number of edge crossings k [17, 18, 25], including sub-exponential algorithms. The two-sided variant of the problem (where the permutation of both sides is variable) is also FPT in the number of crossings [37]. OSCM is a cornerstone of algorithms dealing with the so-called *Sugiyama approach* to hierarchical graph drawing, see [32, 53].

Now, we show that OSCM can be reduced into PCIO, starting with a simple remark.

▶ Remark 13. Isolated vertices in V_2 can be placed anywhere in an optimal ordering of V_2.

From here, we assume that V_2 does not contain any isolated vertices. Similar to [23, 25, 38], we can model OSCM instances as PCIO instances.

▶ **Lemma 14.** *Given an instance (G, τ_1, k) of OSCM, one can construct in polynomial time an equivalent instance $(\iota, \mathfrak{c}, k)$ of PCIO.*

As PCIO has not been formally studied in the literature, let us draw an important consequence from the previous lemma (also see the discussion in the beginning of Section 3.2).

▶ **Corollary 15.** *Positive Completion of an Interval Ordering is NP-complete, even when restricted to instances $(\iota, \mathfrak{c}, k)$ where the arc weights are within the set $\{1, 2, \ldots, 16\}$.*

▶ Remark 16. We can use Theorem 8 to immediately deduce an algorithm for OSCM matching the running time $\mathcal{O}^*(2^{\sqrt{2k}})$ of the best published algorithm for OSCM [38]. We could also use the PCO-kernelization as a kernelization procedure for OSCM.

▶ Remark 17. Çakiroglu *et al.* [11] studied the variation where edges (if existing) have positive weights, and the cost of an edge crossing is obtained by the product of the weights of the crossing edges. This modification (with applications in automatic graph drawing) can also be modeled by PCIO, so that we inherit an $\mathcal{O}^*(2^{\sqrt{2k}})$ algorithm for the standard parameter k.

5 The Kemeny Rank Aggregation Problem

Preference lists are extensively used in social science surveys and voting systems to capture information about choice. Kemeny [36] discussed the problem to combine several preference lists into one, called its aggregation. This approach aims at minimizing the total disagreement (formalized below) between the several input rankings and their aggregation. The idea itself has not only applications in (the theory of) elections in the context of social sciences, say, on a committee, but has also been suggested as a means of designing meta-search engines [19]. It has been also shown by Young and Levenglick [56] that the aggregation method proposed by Kemeny is the only one satisfying a number of natural requirements on such aggregations.

More formally, in Kemeny Rank Aggregation we are given a set Π of rankings (also called *votes*) over a set of alternatives C (also called *candidates*), and a positive integer k, and are asked for a ranking π of C, such that the sum of the *Kendall-Tau distances* (or, KT-distances for short) of π from all the votes, called its *Kemeny score*, is at most k. The ranking π that gives the smallest Kemeny score is called a *Kemeny consensus*. The KT-distance between two rankings π_1 and π_2 is the number of pairs of candidates that are ordered differently in the two rankings and is denoted by KT-dist(π_1, π_2). Hence, if $\pi_1, \pi_2 : [|C|] \to C$, KT-dist$(\pi_1, \pi_2) = |\{(c, c') \in C \times C \mid c <_{\pi_1} c' \wedge c' <_{\pi_2} c\}|$. Observe that the Kendall-Tau distance can be seen as the "bubble sort" distance.

Problem name: Kemeny Rank Aggregation (KRA) **Given:** A list of votes Π over a set of candidates C, a non-negative integer k **Output:** Is there a ranking π on C such that the sum of the KT-distances of π from all the votes is at most k.

Hence, given rankings $\pi_1, \ldots, \pi_m$ of C and a non-negative integer k, the question is if there exists a ranking $\pi : [|C|] \to C$ such that $\sum_{i=1}^{m} \text{KT-dist}(\pi, \pi_i) \leq k$. The problem Kemeny Rank Aggregation is known to be NP-complete [2], even if only four votes are input [19].[3] Simjour [51] obtained an algorithm for the problem that runs in time $\mathcal{O}^*(1.403^k)$. There are also sub-exponential algorithms for Kemeny Rank Aggregation under this parameterization: Karpinski and Schudy [35] obtained an algorithm for Kemeny Rank Aggregation that runs in $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k})})$ time, while the algorithm of Fernau *et al.* [24, 25], based on a different methodology, runs in $\mathcal{O}^*(k^{\mathcal{O}(\sqrt{k})})$ time. Both algorithms hide some constant factor in the $\mathcal{O}$-notation in the exponent that is not that clear from the expositions. Our considerations are also valid for *weighted Kemeny score*, a modification suggested in [5] that assigns positive weights to the voters. We can add some comment on conditional lower bounds of this problem by bringing together facts from different parts of the literature.

[3] The proof of this fact is not contained in the conference paper [19] but only appears in Appendix B of `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/rank_www10.html`.

▶ **Theorem 18.** *KRA on instances with only $m = 4$ votes on some candidate set C and some integer k bounding the sum of the Kendall-Tau distances to a solution cannot be solved neither in time $\mathcal{O}^*\left(2^{o(|C|)}\right)$ nor in time $\mathcal{O}^*\left(2^{o(\sqrt{k})}\right)$, unless ETH fails.*

5.1 Reduction from KRA to PCO

Now we will show that KRA can be encoded into PCO. Let (Π, C) be an instance of Kemeny Rank Aggregation with m votes $\Pi = (\pi_1, \ldots, \pi_m)$ over n candidates C. From this instance, we construct an equivalent instance of the PCO problem $\rho \subseteq V \times V$, $\mathfrak{c}$ with base set $V = C$. For every pair of candidates c_1 and c_2, we define the *cost* of (c_1, c_2), $\mathfrak{c}(c_1, c_2)$, as the number of votes that do not order c_1 before c_2. More formally, $\mathfrak{c}(c_1, c_2) = |\{i \in [m] \mid c_2 <_{\pi_i} c_1\}|$.

▶ **Lemma 19.** *Given two candidates c_1 and c_2, if for every vote $\pi_i \in \Pi$, we have $c_1 <_{\pi_i} c_2$ then for every Kemeny consensus π, $c_1 <_\pi c_2$.*

Using different terminology, a proof of this lemma can be found in [43, Théorème 3]. Now, we define the partial order ρ as follows: $(c_1, c_2) \in \rho$ if and only if $\mathfrak{c}(c_1, c_2) = 0$. Hence, $<_\rho = \bigcap_{i=1}^m <_{\pi_i}$ is the *unanimity order* [12]. By Lemma 19, a vote π, which is a linear order of the candidates, is a Kemeny consensus iff π is a linear extension of ρ of minimum cost with Kemeny score[4]

$$\sum_{i=1}^{m} \text{KT-dist}(\pi, \pi_i) = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{n}[c_j <_{\pi_i} c_k \wedge c_k <_\pi c_j] = \sum_{j=1}^{n}\sum_{k=1}^{n}\mathfrak{c}(c_k, c_j)[c_k <_\pi c_j] \qquad (1)$$

is equal to the cost of the linear extension given by π according to its definition.
These considerations prove that we can translate our algorithmic results for PCO to KRA.

▶ Remark 20. Our reduction works even if votes are reflexive and antisymmetric relations instead of linear orders. In this case, the cost between c_1 and c_2 is defined as follows: $\mathfrak{c}(c_1, c_2) = |\{i \in [m] \mid c_1 \not<_{\pi_i} c_2\}|$.

5.2 Pathwidth in Kemeny Rank Aggregation

Now we will discuss the meaning of the pathwidth measure from the PCO problem applied to KRA. For KRA, several measures have been studied in the context of parameterized complexity, Betzler et al. [5] introduced the notion of *maximum range of candidate positions*. For an election (Π, C), the *range* $r(c)$ of a candidate c is defined as $r(c) \doteq \max_{i,j\in[m]} |\pi_i^{-1}(c) - \pi_j^{-1}(c)| + 1$. If $\Pi(c) \doteq \{i \in [|C|] \colon \exists \pi \in \Pi : \pi(i) = c\}$ denotes the set of positions candidate c received in election (Π, C), then $r(c) = \max \Pi(c) - \min \Pi(c) + 1$. The *maximum range* $r_{\max}$ of an election is given by $r_{\max} \doteq \max_{c\in C} r(c)$. Betzler et al. [5] proved that KRA can be solved in time $\mathcal{O}(32^{r_{\max}} \cdot (r_{\max}^2 \cdot |C| + r_{\max} \cdot |C|^2 \log |C| \cdot m) + m^2 \cdot |C| \log |C|) = \mathcal{O}^*(2^{5r_{\max}})$.

▶ **Lemma 21.** *Given an election (Π, C), let w be the consistent pathwidth associated to the election and $r_{\max}$ be the maximum range of the election. We have $\mathrm{w} \leq 2 \cdot r_{\max} - 2$.*

Proof. Let ρ be the partial order defined by the election. To prove this statement, we will construct an interval order ι such that $\iota \subseteq \rho$, and $\mathrm{w}(\iota) \leq 2 \cdot r_{\max}$. To each candidate $c \in C$, we associate the interval $I_c \doteq [\min \Pi(c) - 1, \max \Pi(c))$. (We subtract one from the left border to avoid empty intervals.) We let ι be the interval order associated with the interval

[4] Recall the bracket notation: if p is a logical proposition, then $[p]$ yields 1 if p is true and else, $[p]$ yields 0.

representation $\{I_c \mid c \in C\}$. By Lemma 19, we have that ρ is an extension of the interval order ι. Each interval I_c has length at most $r_{\max}$. Thus, there are at most $2 \cdot r_{\max} - 2$ intervals that intersect at one point in the interval representation. Hence, $\mathrm{w}(\iota) \leq 2 \cdot r_{\max} - 2$. ◀

Hence, Theorem 1 yields the following noticeable improvement to the mentioned result of [5]:

▶ **Corollary 22.** *KRA can be solved in time* $\mathcal{O}(|C| \cdot r_{\max} \cdot 2^{2r_{\max}} + m \cdot |C|^2) = \mathcal{O}^*(2^{2r_{\max}})$.

6 Grouping by Swapping (GbS)

This problem asks whether a given string can be transformed by at most k interchanges of neighboring letters into a block format where all occurrences of each letter are adjacent to form one single block each. It is on the famous list of NP-complete problems in [28]. Further algorithmic aspects are discussed in [15, 55]. We show that GbS can be reduced to OSCM in a parameter-preserving way and hence inherits FPT-results shown above. We first discuss the problem GbS itself and then continue with the reductions.

Problem name: Grouping by Swapping (GbS)
Given: A finite alphabet Σ, a string $w \in \Sigma^*$, and $k \in \mathbb{N}$.
Output: Is there a sequence of at most k adjacent swaps such that w is transformed into a string w' where all occurrences of each symbol are in single blocks?

Let us formalize this problem a bit more. If $w, w' \in \Sigma^*$ both have length n, we call w' a *permutation* of w if there exists a bijection $\pi : [n] \to [n]$ such that, for any $i \in [n]$, $w'[i] = w[\pi(i)]$. Slightly abusing notation, we will also write $w' = \pi(w)$. Special bijections are adjacent swaps $\sigma_i : [n] \to [n]$ (with $i \in [n-1]$) that act as the identity with two exceptions: $\sigma_i(i) = i+1$ and $\sigma_i(i+1) = i$. Every bijection $\pi : [n] \to [n]$ can be written as a composition of swaps (property (*)). Hence, given a permutation w' of w, we can ask to compute the *swap distance*, written $sd(w, w')$, which is the smallest number k of swaps $\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_k}$ such that $w' = (\sigma_{i_1} \circ \sigma_{i_2} \circ \cdots \circ \sigma_{i_k})(w)$. Observe that sd can be viewed, for each mapping $g : \Sigma \to \mathbb{N}$, as a metric on the space of all words $w \in \Sigma^*$ with $g(a)$ occurrences of a for each letter $a \in \Sigma$. In particular, $sd(w, w') = sd(w', w)$ for all permutations w' of w. Notice that the swap distance can be computed in quadratic time by dynamic programming, as shown in [42] (property (+)).

This picture changes if we add one more degree of freedom. Let us call $w' \in \Sigma^*$ to be in *block format* if there is a bijection $f : [|\Sigma|] \to \Sigma$ such that $w' \in f(1)^* f(2)^* \cdots f(|\Sigma|)^*$. Alternatively, we can view f as defining a linear order $<_f$ on Σ, and then the block format of w corresponding to f is the $<_f$-lexicographic smallest permutation of w. GbS now asks, given $w \in \Sigma^*$ and $k \geq 0$, if there is some permutation w' of w that is in block format and has swap distance at most k from w. As claimed in [28], this variant is NP-complete. Unfortunately, the proof referenced by [28] is hidden in a private communication. We remedy this below by proving that GbS is NP-complete even for strings w where each letter occurs exactly four times. Let us start with two rather straightforward observations.

▶ **Lemma 23.** *Any string w can be grouped into blocks using at most $|w|^2$ many swaps.*

In fact, any permutation of w can be obtained by using at most $|w|^2$ many swaps, as can be seen by bubble-sort. This reasoning also shows (*), a well as (+), with a little bit of thinking. This can be used to obtain our first (easy) FPT-result, to be improved on later.

▶ **Lemma 24.** *GbS on strings $w \in \Sigma^n$ parameterized by $|\Sigma|$ can be solved in time $\mathcal{O}^*(|\Sigma|!)$.*

We are now going to show that computing the swap distance can be done by considering the distance for pairs of letters, summing up the corresponding results. Notice that the formula in Lemma 25 resembles earlier derived summation formulae, as the defining equation for PCO. To make this more precise, let $\Sigma' \subseteq \Sigma$ and consider the projection $p_{\Sigma,\Sigma'} : \Sigma \to \Sigma'$ that maps $a \mapsto a$ for $a \in \Sigma'$ and $a \mapsto \varepsilon$, the empty word, if $a \notin \Sigma'$, as a morphism $\Sigma^* \to (\Sigma')^*$.

▶ **Lemma 25.** *Let $w, w' \in \Sigma^*$ such that w' is a permutation of w. Let w' be in block format following the linear order τ on Σ. $sd(w, w') = \sum_{a,b \in \Sigma, a <_\tau b} sd(p_{\Sigma,\{a,b\}}(w), p_{\Sigma,\{a,b\}}(w'))$. Moreover, $p_{\Sigma,\{a,b\}}(w') = a^{|w|_a} b^{|w|_b}$ if $a <_\tau b$.*

6.1 Discussing NP-completeness

In this subsection, we will prove NP-completeness of GbS even for quite restricted instances by making use of a somewhat similar result for OSCM, based on [44].

▶ **Theorem 26.** *GbS is NP-complete, even if each letter has exactly 4 occurrences.*

Proof. Membership in NP is clear. In order to show NP-hardness, we give a reduction from OSCM which is also NP-complete if each node in V_2 has degree four and each vertex in V_1 has degree one, i.e., if the graph is a forest of 4-stars [44], with all star's centers in V_2. Let $G = (V_1, V_2, E)$ be an instance of OSCM with order τ_1 on V_1 and integer k such that all vertices in V_1 are of degree one and all vertices in V_2 are of degree four. We set $\Sigma = V_2 = \{v_1, v_2, \ldots, v_n\}$. Clearly, $|V_1| = 4n$. We construct $w \in \Sigma^{4n}$ (starting from the empty word) by going through the vertices in V_1, following the order τ_1. If the current vertex is adjacent to v_i, we concatenate v_i to w. As the vertices in V_1 are of degree one, this assignment is unambiguous. Following [21], for vertices $v_i, v_j \in V_2$, let $c_{v_i v_j}$ be the number of crossings between edges incident to v_i and edges incident to v_j when v_i is placed left of v_j. Lemma 3 in [21] states, referring to [20], that for a linear order τ_2 on V_2, the number of crossings $cross(G, \tau_1, \tau_2)$ of the edges between V_1 in order τ_1 and V_2 in order τ_2 is $cross(G, \tau_1, \tau_2) = \sum_{v_i, v_j \in V_2, v_i <_{\tau_2} v_j} c_{v_i v_j}$. Clearly, for $v_i, v_j \in V_2$ the number of crossings $c_{v_i v_j}$ is equal to $sd(p_{\Sigma,\{v_i,v_j\}}(w), p_{\Sigma,\{v_i,v_j\}}(w_{\tau_2}))$, where w_{τ_2} is the τ_2-lexicographic smallest permutation of w. Combining this observation with Lemma 25, we obtain that for every linear order τ_2, $sd(w, w_{\tau_2}) = cross(G, \tau_1, \tau_2)$. ◀

In the following subsection, we will show that, in a sense, the reduction presented in our NP-hardness result for GbS can be reversed. This also shows the following:

▶ Remark 27. GbS is polynomial-time solvable when each letter occurs at most twice.

This also leaves the following case as an **open question**: Can GbS instances be solved in polynomial time if each letter occurs at most thrice? Notice that it is also open whether subcubic OSCM graph instances can be solved in polynomial time. Furthermore, within KRA, it is open if instances with three voters can be solved in polynomial time. Also, Cor. 15 leaves some room for improvement.

6.2 Reduction from GbS to OSCM

With the same idea as in the proof of Theorem 26, we can also reduce GbS to OSCM by representing the string w as the ordered vertex set V_1 and Σ as the vertex set V_2. More precisely, let n be the length of w and interpret w as a mapping from $[n]$ into Σ. Moreover, set $V_1 = [n]$ with the usual linear ordering $<_{\tau_1} \doteq <$ on $[n]$. Let $V_2 = \Sigma$ and connect $a \in V_2$ to $i \in [n]$ iff $w(i) = a$. This defines the bipartite graph $G = (V_1, V_2, E)$ with linear ordering

τ_1 on V_1. Now, the GBS instance (w, k) is a YES-instance if and only if the constructed OSCM instance (G, τ_1, k) is a YES-instance. As OSCM is solvable in polynomial time if the vertices in V_2 have degree at most two, this implies that GBS is solvable in polynomial time if each letter has at most two appearances, see Remark 27 and the following comments.

Together with the reduction proving Theorem 26, we see that GBS can be viewed as exactly the special case of OSCM where all vertices of V_1 are of degree one, so that the instance becomes a forest of stars with centers in V_2. We make the algorithmic consequences of this connection explicit, each time giving references to the literature on OSCM.

▶ **Corollary 28.** *GBS, parameterized by k, can be solved (in polynomial space) in time $\mathcal{O}^*(1.4656^k)$ by [17] or (in exponential space) in time $\mathcal{O}^*(2^{\sqrt{2k}})$ by [38] or Remark 16.*

Fernau *et al.* [25] and Kobayashi and Tamaki [38] also obtained OSCM lower bound results, based on [44], assuming ETH. By the proof of Theorem 26, we can strengthen them:

▶ **Corollary 29.** *GBS on strings of length n over alphabet Σ, parameterized by the number k of swaps, cannot be solved neither in time $\mathcal{O}^*(2^{o(n)})$ nor in time $\mathcal{O}^*(2^{o(|\Sigma|)})$ nor in time $\mathcal{O}^*(2^{o(\sqrt{k})})$, unless ETH fails, even if each letter has exactly 4 occurrences.*

6.3 Pathwidth in Grouping by Swapping

The concept of scope coincidence degree (SCD for short) was introduced in [49] for patterns, which are strings over two disjoint alphabets, where only the alphabet of variables was used to measure the SCD of patterns. We adapt it in the following to strings over a single alphabet.

Given a string $w \in \Sigma^*$, and a letter $a \in \Sigma$, then the *scope* of a, denoted $Scope(a)$ is the set of positions in $\{1, \ldots, |w|\}$ between the minimum position and the maximum position in which a occurs. For each position i, we let the incidence set of i to be $Inc(i) = \{a \in \Sigma \ : \ i \in Scope(a)\}$. Now the scope coincidence degree is the number of overlapping scopes for all letters. In other words, we have that $SCD(w) = \max_i |Inc(i)|$.

Our reduction from GBS to OSCM first turns $w \in \Sigma^*$ into a bipartite graph $G = (V_1, V_2, E)$ with $V_1 = [|w|]$ and $V_2 = \Sigma$. Lemmas 14 then produce an equivalent PCIO-instance with an associated partial order ρ_w on V_2 that is an interval order. For two letters $a, b \in \Sigma$, $(a, b) \in \rho_w$ means that the last occurrence of a in w comes before the first occurrence of b in w. Obviously, $SCD(w)$ is the maximum size of an anti-chain in ρ_w. Hence, the previously mentioned results of Habib and Möhring imply, together with Lemma 5:

▶ **Lemma 30.** $SCD(w) = pw(G_{\rho_w}) + 1 = cpw(G_{\rho_w}, \rho_w) + 1$.

Theorem 1 has therefore the following consequences for the string parameter *SCD*. To the best of our knowledge, this is the first algorithmic exploit of this string parameter.

▶ **Corollary 31.** *GBS can be solved in $\mathcal{O}^*(SCD(w)2^{SCD(w)})$.*

As the scope coincidence degree of a word $w \in \Sigma^*$ is upper-bounded by $|\Sigma|$, we also obtain the following result for the parameter $|\Sigma|$ that improves on Lemma 24.

▶ **Corollary 32.** *GBS can be solved in $\mathcal{O}^*(|\Sigma|2^{|\Sigma|})$.*

There is another graph-theoretic interpretation of the scope coincidence degree presented by Reidenbach and Schmid [49] for patterns. It relates to our setting as follows. To a string $w \in \Sigma^n$, we associate its *Gaifman graph* Γ_w with vertex set $[n]$ and edges $(i, i+1)$ for $i \in [n-1]$, as well as the edge sets $E_a = \{(\min Scope(a), j) \mid j \in Scope(a)\}$ (disregarding loops) for each $a \in \Sigma$. According to [49, Lemma 15], $pw(\Gamma_w) \leq SCD(w) + 1$. It might be interesting to further link the pathwidths of Γ_w and of G_{ρ_w}. Do they differ by exactly two?

Inspired by the considerations on the *range of a candidate* in KRA, the *maximum scope* $s_{\max} \doteq \max_{a \in \Sigma} |Scope(a)|$ could be another parameterization for GBS. Similar to Lemma 21, one can show that GBS, parameterized by $s_{\max}$, is in FPT. It would also be meaningful to interpret this parameter in the context of OSCM for graph visualization reasons.

7 Conclusion

Finally, we explain some further connections and future lines of research. Recall that we did list several concrete open problems throughout the paper that we are not going to repeat here, but they are clearly also natural continuations of the present study.

Different types of partial orderings. It would be interesting to have a closer look to different types of partial orderings in the context of PCO. For instance, the papers of Brandenburg and Gleißer [8] or Hudry [34] list quite a lot of different types of partial orders (in the context of rank aggregation problems). We can also view this research as a starting point to systematically look at decision problems related to partial orders from the viewpoint of parameterized complexity. Then, [7] might be a good starting point.

Related problems, popular with Operations Research. In the Operations Research Community, there has also been lots of studies of the *linear ordering polytope*. Regarding the problems studied in this paper, [10] might be a good starting point. Likewise, the so-called Optimal Linear Extension Problem has been considered in the literature [41]. However, only the costs of the immediate neighborhood in the target linear order are considered, similar to the famous Travelling Salesperson Problem,[5] while we sum up all costs associated to pairs (x, y) with $x < y$ in the final linear order $<$.

Putting additional constraints: a theme arising in Graph Drawing and in Order Theory. Forster [27] argues that the Constrained OSCM problem, where a partial order on V_2 is given in addition, that should be extended to a linear ordering (as before), has quite some applications. This can be clearly modeled as an instance of CO, but some further research is needed to conclude the same type of results as we did for OSCM with the interval order approach. This might relate to earlier (systematic) research on the realizability of constraints on interval orders, see [47, 48]. In particular the distance constraints might be indeed interesting for graph drawing purposes, as the neighbor vertices should not stretch out too much.

Remarks on approximation. For the minimization problem related to PCO, a PTAS is known according to [25]. Our reasoning immediately implies the existence of PTAS for OSCM, KRA and GBS. In view of the tedious factor-1.4664 approximation for OSCM presented in [46], this shows again the strength of looking at these specific problems from a wider perspective.

Comments on approximation and heuristics. We suggest that the tight connections that we found between GBS and OSCM should also be interesting in the development and analysis of (heuristic) algorithms for both problems. In this context, it is interesting to observe that

[5] The difference between cycles (tours) and paths do not matter for the involved algorithms that much.

Wong and Reingold [55] proposed a median heuristic for computing a solution to a given GBS instance. They proved that on random instances, this heuristic is at most 10% off from the optimum (in expectation). Moreover, the larger random instances are picked, the smaller is the relative error of the median heuristic (in expectation). Incidentally, the same (median) heuristic was suggested by Eades and Wormalds [21] some years later for OSCM. They proved that this heuristic is a factor-3 approximation, but did not go into a randomized analysis. Our translation of GBS into OSCM actually proves the following which is the last result of this paper.

▶ **Corollary 33.** *The median heuristic gives a factor-3 approximation for GBS.*

References

1 Noga Alon, Daniel Lokshtanov, and Saket Saurabh. Fast FAST. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of *Lecture Notes in Computer Science*, pages 49–58. Springer, 2009. doi:10.1007/978-3-642-02927-1_6.

2 J. Bartholdi, III, C. A. Tovey, and M. A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6:157–165, 1989.

3 Oliver Bastert and Christian Matuszewski. Layered drawings of digraphs. In Michael Kaufmann and Dorothea Wagner, editors, *Drawing Graphs, Methods and Models (the book grow out of a Dagstuhl Seminar, April 1999)*, volume 2025 of *Lecture Notes in Computer Science*, pages 87–120. Springer, 1999. doi:10.1007/3-540-44969-8_5.

4 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

5 Nadja Betzler, Michael R. Fellows, Jiong Guo, Rolf Niedermeier, and Frances A. Rosamond. Fixed-parameter algorithms for kemeny rankings. *Theor. Comput. Sci.*, 410(45):4554–4570, 2009. doi:10.1016/j.tcs.2009.08.033.

6 Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13(3):335–379, 1976. doi:10.1016/S0022-0000(76)80045-1.

7 Vincent Bouchitté and Michel Habib. NP-completeness properties about linear extensions. *Order*, 4(2):143–154, 1987.

8 Franz J. Brandenburg and Andreas Gleißner. Ranking chain sum orders. *Theor. Comput. Sci.*, 636:66–76, 2016. doi:10.1016/j.tcs.2016.05.026.

9 Franz-Josef Brandenburg, Andreas Gleißner, and Andreas Hofmeier. Comparing and aggregating partial orders with kendall tau distances. *Discrete Math., Alg. and Appl.*, 5(2), 2013. doi:10.1142/S1793830913600033.

10 Christoph Buchheim, Angelika Wiegele, and Lanbo Zheng. Exact algorithms for the quadratic linear ordering problem. *INFORMS J. Comput.*, 22(1):168–177, 2010. doi:10.1287/ijoc.1090.0318.

11 Olca A. Çakiroglu, Cesim Erten, Ömer Karatas, and Melih Sözdinler. Crossing minimization in weighted bipartite graphs. *J. Discrete Algorithms*, 7(4):439–452, 2009. doi:10.1016/j.jda.2008.08.003.

12 Irène Charon and Olivier Hudry. A survey on the linear ordering problem for weighted or unweighted tournaments. *4OR*, 5(1):5–60, 2007. doi:10.1007/s10288-007-0036-6.

13 Derek G. Corneil, Barnaby Dalton, and Michel Habib. LDFS-based certifying algorithm for the minimum path cover problem on cocomparability graphs. *SIAM J. Comput.*, 42(3):792–807, 2013. doi:10.1137/11083856X.

14 Derek G. Corneil, Stephan Olariu, and Lorna Stewart. The LBFS structure and recognition of interval graphs. *SIAM J. Discret. Math.*, 23(4):1905–1953, 2009. doi:10.1137/S0895480100373455.

15 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

16 Vida Dujmovic, Henning Fernau, and Michael Kaufmann. Fixed parameter algorithms for one-sided crossing minimization revisited. In Giuseppe Liotta, editor, *Graph Drawing, 11th International Symposium, GD 2003, Perugia, Italy, September 21-24, 2003, Revised Papers*, volume 2912 of *Lecture Notes in Computer Science*, pages 332–344. Springer, 2003. doi:10.1007/b94919.

17 Vida Dujmovic, Henning Fernau, and Michael Kaufmann. Fixed parameter algorithms for one-sided crossing minimization revisited. *J. Discrete Algorithms*, 6(2):313–323, 2008.

18 Vida Dujmovic and Sue Whitesides. An efficient fixed parameter tractable algorithm for 1-sided crossing minimization. *Algorithmica*, 40(1):15–31, 2004.

19 Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In Vincent Y. Shen, Nobuo Saito, Michael R. Lyu, and Mary Ellen Zurko, editors, *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 613–622. ACM, 2001. doi:10.1145/371920.372165.

20 Peter Eades and D. Kelly. Heuristics for reducing crossings in 2-layered networks. *Ars Combinatoria*, 21A:89–98, 1986.

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

22 Eduard Eiben, Robert Ganian, Kustaa Kangas, and Sebastian Ordyniak. Counting linear extensions: Parameterizations by treewidth. *Algorithmica*, 81(4):1657–1683, 2019. doi:10.1007/s00453-018-0496-4.

23 Henning Fernau. *Parameterized Algorithmics: A Graph-Theoretic Approach*. Habilitationsschrift, Universität Tübingen, Germany, 2005.

24 Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Matthias Mnich, Geevarghese Philip, and Saket Saurabh. Ranking and drawing in subexponential time. In Costas S. Iliopoulos and William F. Smyth, editors, *Combinatorial Algorithms - 21st International Workshop, IWOCA 2010, London, UK, July 26-28, 2010, Revised Selected Papers*, volume 6460 of *Lecture Notes in Computer Science*, pages 337–348. Springer, 2011.

25 Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Matthias Mnich, Geevarghese Philip, and Saket Saurabh. Social choice meets graph drawing: How to get subexponential time algorithms for ranking and drawing problems. *TSINGHUA SCIENCE AND TECHNOLOGY*, 19(4):374–386, 2014.

26 Peter C. Fishburn. *Interval Orders and Interval Graphs*. John Wiley, 1985.

27 Michael Forster. A fast and simple heuristic for constrained two-level crossing reduction. In János Pach, editor, *Graph Drawing, 12th International Symposium, GD 2004, New York, NY, USA, September 29 - October 2, 2004, Revised Selected Papers*, volume 3383 of *Lecture Notes in Computer Science*, pages 206–216. Springer, 2004. doi:10.1007/978-3-540-31843-9_22.

28 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

29 Paul C Gilmore and Alan J Hoffman. A characterization of comparability graphs and of interval graphs. *Canadian Journal of Mathematics*, 16:539–548, 1964.

30 Michel Habib, Ross M. McConnell, Christophe Paul, and Laurent Viennot. Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theor. Comput. Sci.*, 234(1-2):59–84, 2000. doi:10.1016/S0304-3975(97)00241-7.

31 Michel Habib and Rolf H. Möhring. Treewidth of cocomparability graphs and a new order-theoretic parameter. *Order*, 11(1):47–60, 1994.

32 Patrick Healy and Nikola S. Nikolov. Hierarchical drawing algorithms. In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization*, pages 409–453. Chapman and Hall/CRC, 2013.

33 Wen-Lian Hsu and Tze-Heng Ma. Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. *SIAM J. Comput.*, 28(3):1004–1020, 1999. doi:10.1137/S0097539792224814.

34 Olivier Hudry. NP-hardness results for the aggregation of linear orders into median orders. *Annals of Operations Research*, 163:63–88, 2008.

35 Marek Karpinski and Warren Schudy. Faster algorithms for feedback arc set tournament, Kemeny rank aggregation and betweenness tournament. In Otfried Cheong, Kyung-Yong Chwa, and Kunsoo Park, editors, *Algorithms and Computation - 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I*, volume 6506 of *Lecture Notes in Computer Science*, pages 3–14. Springer, 2010.

36 John G. Kemeny. Mathematics without numbers. *Daedalus*, 88:571–591, 1959.

37 Yasuaki Kobayashi, Hirokazu Maruta, Yusuke Nakae, and Hisao Tamaki. A linear edge kernel for two-layer crossing minimization. *Theor. Comput. Sci.*, 554:74–81, 2014.

38 Yasuaki Kobayashi and Hisao Tamaki. A fast and simple subexponential fixed parameter algorithm for one-sided crossing minimization. *Algorithmica*, 72(3):778–790, 2015.

39 Johannes Köbler, Sebastian Kuhnert, Bastian Laubner, and Oleg Verbitsky. Interval graphs: Canonical representations in logspace. *SIAM J. Comput.*, 40(5):1292–1315, 2011. doi:10.1137/10080395X.

40 Johannes Köbler, Sebastian Kuhnert, and Osamu Watanabe. Interval graph representation with given interval and intersection lengths. *J. Discrete Algorithms*, 34:108–117, 2015. doi:10.1016/j.jda.2015.05.011.

41 Longcheng Liu, Biao Wu, and Enyu Yao. Minimizing the sum cost in linear extensions of a poset. *J. Comb. Optim.*, 21(2):247–253, 2011. doi:10.1007/s10878-009-9237-6.

42 Roy Lowrance and Robert A. Wagner. An extension of the string-to-string correction problem. *J. ACM*, 22(2):177–183, 1975. doi:10.1145/321879.321880.

43 B. Monjardet. Tournois et ordres médians pour une opinion. *Mathématiques et Sciences humaines*, 43:55–73, 1973.

44 Xavier Muñoz, Walter Unger, and Imrich Vrt'o. One sided crossing minimization is NP-hard for sparse graphs. In Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors, *Graph Drawing, 9th International Symposium, GD 2001 Vienna, Austria, September 23-26, 2001, Revised Papers*, volume 2265 of *Lecture Notes in Computer Science*, pages 115–123. Springer, 2001. doi:10.1007/3-540-45848-4.

45 Petra Mutzel. Optimization in leveled graphs. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization, Second Edition*, pages 2813–2820. Springer, 2009. doi:10.1007/978-0-387-74759-0_483.

46 Hiroshi Nagamochi. An improved bound on the one-sided minimum crossing number in two-layered drawings. *Discret. Comput. Geom.*, 33(4):569–591, 2005. doi:10.1007/s00454-005-1168-0.

47 Itsik Pe'er and Ron Shamir. Realizing interval graphs with size and distance constraints. *SIAM J. Discret. Math.*, 10(4):662–687, 1997. doi:10.1137/S0895480196306373.

48 Itsik Pe'er and Ron Shamir. Satisfiability problems on intervals and unit intervals. *Theor. Comput. Sci.*, 175(2):349–372, 1997. doi:10.1016/S0304-3975(96)00208-3.

49 Daniel Reidenbach and Markus L. Schmid. Patterns with bounded treewidth. *Inf. Comput.*, 239:87–99, 2014.

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

51 Narges Simjour. Improved parameterized algorithms for the Kemeny aggregation problem. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of *Lecture Notes in Computer Science*, pages 312–323. Springer, 2009.

52 Matthias F. M. Stallmann, Franc Brglez, and Debabrata Ghosh. Heuristics, experimental subjects, and treatment evaluation in bigraph crossing minimization. *ACM J. Exp. Algorithmics*, 6:8, 2001. doi:10.1145/945394.945402.

53 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst. Man Cybern.*, 11(2):109–125, 1981. doi:10.1109/TSMC.1981.4308636.

54 William T. Trotter. New perspectives on interval orders and interval graphs. In R. A. Bailey, editor, *Surveys in Combinatorics*, volume 241 of *London Mathematical Society Lecture Note Series*, pages 237–286. 1997.

55 D. F. Wong and Edward M. Reingold. Probabilistic analysis of a grouping algorithm. *Algorithmica*, 6(2):192–206, 1991. doi:10.1007/BF01759041.

56 H. P. Young and A. Levenglick. A consistent extension of Condercet's election principle. *SIAM J. Appl. Math.*, 35(2):285–300, 1978.

57 Anke van Zuylen and David P. Williamson. Deterministic pivoting algorithms for constrained ranking and clustering problems. *Math. Oper. Res.*, 34(3):594–620, 2009. doi:10.1287/moor.1090.0385.