ELTE Eötvös Loránd University

Faculty of Science

Linear extensions of partially ordered sets

Math project II. report

Júlia Éles

Applied mathematics MSc

Supervisor:

Péter Madarasi

HUN-REN Alfréd Rényi Institute of Mathematics, and
Department of Operations Research, Eötvös Loránd University

Budapest
2026

1 Introduction

Given a set P and a partial ordering $\prec$ on it. Let us count the number of *linear extensions* of the partial order, i.e., the number of distinct bijective functions $\pi : P \to \{1, \ldots, n\}$ such that $\forall a, b \in P : a \prec b \Rightarrow \pi(a) < \pi(b)$, where n is the number of elements in the set P. The task is equivalent to determining the number of topological orderings of the *cover graph* $G = (V, A)$ of the partial order, where $V = P$ and $A = \{uv \in P^2 : u \prec v, \nexists w \in P \setminus \{u, v\} : u \prec w \prec v\}$.

This problem has been studied extensively: G. Brightwell and P. Winkler proved in 1991 that the problem is #P-hard [2]. In 2016, Kangas et al. provided a polynomial algorithm for partial orderings whose cover graph has constant treewidth; their dynamic programming-based algorithm has running time $O(n^{t+4})$, where t is the treewidth of the graph [4]. This was later improved by K. Kangas, M. Koivisto, and S. Salonen to $O(n^{t+3})$ time in 2020 using a nice tree decomposition of the graph [5].

2 Counting linear extensions in trees

2.1 The concept of spectrum and its calculation

In general, it is difficult to determine the number of topological orderings, but if the cover graph is a tree, there are solutions that are faster than the algorithms mentioned above. M. D. Atkinson presented an algorithm in 1990 that performs $O(n^2)$ arithmetic operations [1].

The algorithm is based on *spectra*. The *spectrum function* associated with the graph G is the function $\sigma_G : V \to \mathbb{Z}_+^n$. The *spectrum* of the vertex $v \in V$ is $\sigma_G(v) = (\sigma_G(v)_1, \sigma_G(v)_2, \ldots, \sigma_G(v)_n)$, where $\sigma_G(v)_i$ is the number of topological orderings in which the vertex v is in the i-th position.

In one step of the algorithm, we split the graph along an arc uv. Let G_u be the component containing u after the split, and G_v be the component containing v. Then, using $\sigma_{G_u}(u)$ and $\sigma_{G_v}(v)$, we can easily determine the value of either $\sigma_G(u)$ or $\sigma_G(v)$. The values of $\sigma_{G_u}(u)$ and $\sigma_{G_v}(v)$ can be calculated recursively. It follows that the number of topological orderings of the graph is $\sum_{i=1}^{n} \sigma_G(v)_i$ for any vertex v.

2.2 Spectrum for two vertices, arc spectrum

The *common spectrum function* is a generalization of the previously defined spectrum function, as we specify the positions of two vertices instead of one. Formally, the common spectrum function with respect to the graph G is the function $\tau_G : V^2 \to \mathbb{Z}_+^{n \times n}$, where the common spectrum of a vertex pair (u, v) is a matrix $\tau_G(u, v)$ in which $\tau_G(u, v)_{ij}$ is the number of topological orderings where u is in the i-th position and v is in the j-th position.

During previous semesters, we attempted to find a method for determining the aforementioned common spectra. We were successful with a simpler version, namely *arc spectra*, which are a specific case of common spectra: u and v are connected by an arc in the graph, i.e., $uv \in A$.

To calculate the arc spectrum of u and v, we need the spectra of u and v. We obtain them by cutting the graph along the arc uv, where G_u is the component containing u, and G_v is the component containing the vertex v, and then determining the values of $\sigma_{G_u}(u)$ and $\sigma_{G_v}(v)$ using the above algorithm. Let p denote the number of vertices in G_u, q denote the number of vertices in G_v, and n denote the number of vertices in G (we know that $p + q = n$).

We want to determine $\tau_G(u, v)_{rs}$. We know how many orders there are in G_u where vertex u is in the i-th position, and how many orders there are in G_v where vertex v is in the j-th position. We want to merge two such sequences. There are already $i - 1$ elements before vertex u, which means that $r - i$ elements must still be selected from sequence of v; these elements can be freely "mixed". Similarly, there are already $q - j$ elements after vertex v, so $p - s + j$ elements from the other sequence must be placed after v. Thus, our combined sequence is divided into three parts by u and v, in which the elements in the

sequences u and v are mixed. After a quick calculation, we get the following formula:

$$\tau_G(u,v)_{rs} = \sum_{i=1}^{\min(r,p)} \binom{r-1}{i-1} \sigma_{G_u}(u)_i \cdot \sum_{j=\max(r-i+1,s-p)}^{\min(q,s-i)} \binom{s-r-1}{s-j-i}\binom{n-s}{q-j} \cdot \sigma_{G_v}(v)_j.$$

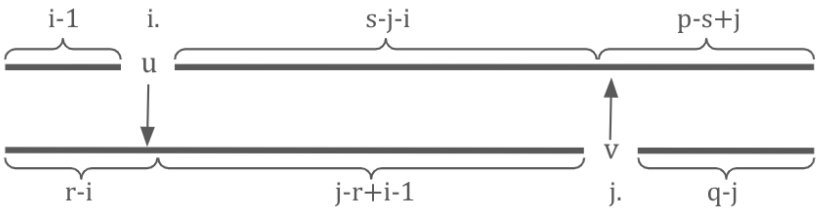

Figure 1: Determining the arc spectrum of u and v from the spectra of u and v

It is also worthwhile to investigate the common spectrum of vertex pairs between which there is a directed path. In such cases, we divide the graph along the arcs of the path, so each vertex will have its own subgraph in which we can compute its spectrum. Then, from these sub-spectra, we can calculate the common spectrum in two ways:

1. We calculate the common spectrum of the first and last elements, excluding the intermediate elements. Taking the intermediate elements into account, we recursively calculate the common spectrum of the two extreme elements. From the two spectra obtained in this way, we calculate the common spectrum of the extreme elements.

2. We calculate the common spectrum of the first and second elements, then calculate the spectrum of the first and third elements from this and the individual spectrum of the third element. We continue this until we reach the other end of the path.

In this semester, we successfully determined the common spectrum of two vertices that are not connected by a directed path. Let the undirected path connecting the two vertices be $u = v_0, v_1, \ldots, v_k = v$, and let us cut the graph along this path. Let the resulting subgraphs be $G_0, G_1, \ldots, G_k$ and let $G'_i \coloneqq \bigcup_{j=0}^{i} G_j$. Suppose that for a given $i \in \{1, \ldots, k-1\}$, we have determined $\tau_{G'_i}(v_0, v_i)_{i_1 i_2}$ and $\sigma_{G_{i+1}}(v_{i+1})_j$, and we want to determine $\tau_{G'_{i+1}}(v_0, v_{i+1})_{rs}$. The method is similar to determining the arc spectrum, but we must handle the cases $r < s$ and $s < r$ separately, and within those, the cases $i_1 < i_2$ and $i_2 < i_1$ as well.

Suppose that $r < s$ and $i_1 < i_2$. Then, $r - i_1$ vertices from the sequence of v_{i+1} must still be placed before v_0, and similarly, $p-s+j$ vertices from the common sequence of v_0 and v_i must be placed after v_{i+1}, where $p \coloneqq |V(G'_i)|$. However, we must pay attention to where v_i is placed relative to v_{i+1}: if $i_2 < s-j+1$, then v_i comes before v_{i+1}, otherwise it comes after. Therefore, if $v_i \prec v_{i+1}$, we can only consider spectra $\tau_{G'_i}(v_0, v_i)_{i_1 i_2}$ where $i_2 < s - j + 1$; otherwise, we consider those for which $i_2 \geq s - j + 1$. Now suppose that $i_2 < i_1$. In this case, we proceed similarly to the $i_1 < i_2$ case, but here v_i will definitely come before v_{i+1}; thus, if $v_{i+1} \prec v_i$, then $\tau_{G'_{i+1}}(v_0, v_{i+1})_{rs} = 0$.

In the case where $r > s$, we proceed in the exact same way.

2.3 Spectrum for three vertices, further conjectures

Our goal in this semester was to extend the concept of spectrum to multiple vertices. The *common spectrum function of k vertices* with respect to the graph G is the function $\tau_G^k : V^k \to \mathbb{Z}_+^{\{1,\ldots,n\}^k}$, where $\tau_G^k(v_1, \ldots, v_k)_{i_1 \ldots i_k}$ is the number of topological orderings where v_j is in the i_j-th position for all $j \in \{1, \ldots, k\}$. Then $\tau_G^2 \equiv \tau_G$.

We believe (though we have not yet been able to verify) that the reduction used for two vertices also works for three vertices v_1, v_2, v_3 that lie on an undirected path: let us determine the common spectrum

of v_1 and v_2, and then sequentially add the vertices along the undirected path $v_2 \backsim v_3$. A similar method works when v_1, v_2, and v_3 do not lie on a single undirected path: take the path between v_1 and v_2, and let u be the vertex on the path from which the branch leading to v_3 attaches. Let w be the vertex following u on this branch, and let us cut the graph along the arc uw. Let the subgraph containing u be G_1, and the subgraph containing w be G_2. Then the values of $\tau^3_{G_1}(v_1, u, v_2)$ and $\tau_{G_2}(w, v_3)$ can be determined, and from these, $\tau^3_G(v_1, v_2, v_3)$ can be computed as well. In both cases, an additional summation appears compared to the formula for the two-vertex spectrum seen earlier. Furthermore, we believe that using this method, the function τ^k_G can be determined for any constant k.

3 Existence of a position-specific ordering

In order to better understand the computation of common spectra for an arbitrary set of k vertices, it is worthwhile to examine the associated decision problem: whether there exists a topological ordering satisfying prescribed position constraints. For this reason, we also investigated the problem domain regarding the existence of a position-specific ordering during the semester.

The task is as follows: given a directed acyclic graph $G = (V, A)$ and a function $f : V \to 2^{\{1,\dots,n\}}$. Here, $f(v)$ is the *set of allowed positions* for vertex v. Determine whether there exists a topological ordering $\pi : V \to \{1, \dots, n\}$ of the graph such that $\pi(v) \in f(v)$ for every vertex $v \in V$ — that is, the position of every vertex falls within its allowed set of positions.

We primarily focus on the case where the set of allowed position is a singleton for a few vertices (i.e., we fix them), while the allowed position set for all other vertices is the set $\{1, \dots, n\}$. The spectrum-based algorithm described above provides a method if the cover graph is a tree.

3.1 Greedy algorithm in linear time when one position is prescribed for certain vertices

An efficient greedy algorithm is already known for the case where one position is allowed for certain vertices and all positions are allowed for the remaining vertices. The algorithm of T. Hagerup and M. Maas from 1994 [3] is based on the following idea: let us determine an interval $[\mathrm{Low}(v), \mathrm{High}(v)]$ for each vertex v, where $\mathrm{Low}(v)$ is a position before which the given vertex cannot be placed, and $\mathrm{High}(v)$ is a position after which the vertex can no longer be placed. These values are determined using the longest directed paths ending at and starting from the given vertex.

The algorithm then iterates through the vertices in increasing order of their High values and attempts to place the vertices into the earliest possible position. Formally: let J be the set of positions reachable by vertex v: $J \coloneqq [\mathrm{Low}(v), \mathrm{High}(v)] \setminus S$, where S is the set of already occupied positions. We place vertex v into position $\pi(v) \coloneqq \min J$ if J is not empty. T. Hagerup and M. Maas proved that if the algorithm does not get stuck (meaning J is never empty), then a sequence satisfying the prescriptions exists.

Their proof was based on the fact that this algorithm always yields a *regular matching* between the vertices and the positions, where a regular matching means the following: for all vertices $u, v \in V$, if v has been placed (i.e., $\pi(v)$ is defined), and $\mathrm{Low}(u) \le \pi(v) \le \mathrm{High}(u) < \mathrm{High}(v)$, then u has also been placed, and $\pi(u) < \pi(v)$. It is easy to see that such a regular matching also defines a correct ordering.

During the semester, we demonstrated that the algorithm also works if the prescriptions are intervals (i.e., $f(v)$ is an interval for every $v \in V$). The algorithm yields a regular matching in this case as well: let $u, v \in V$ be vertices such that $\pi(v)$ is defined and $\mathrm{Low}(u) \le \pi(v) \le \mathrm{High}(u) < \mathrm{High}(v)$. Then we know that u was processed before v. Since $\mathrm{Low}(u) \le \pi(v) \le \mathrm{High}(u)$, it follows that $\pi(v) \in J$ when u was processed, so J was not empty, meaning $\pi(u)$ is also defined. Furthermore, $\pi(u) = \min J \neq \pi(v)$, so $\pi(u) < \pi(v)$.

We verified via a computer program that if the position prescriptions are not intervals, the above greedy algorithm does not work.

3.2 Our own greedy algorithm

During the semester, we developed an alternative algorithm for the task. In the first step of the algorithm, we similarly determine an interval within which the vertex can be placed; however, instead of processing the vertices, we process the positions. It is clear that vertices for which the prescription contains a single element can be fixed to their prescribed positions, so we do not need to deal with them later. We will refer to these as *fixed vertices* in the following, and the positions belonging to them as *fixed positions*. The other vertices and positions will be called *free vertices* and *free positions*, respectively.

Formally, let $l(u) \coloneqq |\{v : \exists v \rightsquigarrow u \text{ path and } v \text{ is a free vertex}\}|$ be the lower bound and $h(u) \coloneqq |\{v : \exists u \rightsquigarrow v \text{ path and } v \text{ is a free vertex}\}|$ be the upper bound. These two bounds indicate how many free positions must be present at least before and after the vertex. Furthermore, let us define the distance between two vertices as follows:

$$d(u,v) \coloneqq \begin{cases} \sum_{w \in V} [1 | w \text{ is free and is an internal vertex of some } u \rightsquigarrow v \text{ path}] & \text{if } \exists u \rightsquigarrow v \text{ path} \\ \text{undefined} & \text{if } \nexists u \rightsquigarrow v \text{ path.} \end{cases}$$

$d(u,v)$ indicates how many free positions must be between u and v if a path $u \rightsquigarrow v$ exists.

For each vertex v, we determine the narrowest set of positions I_v onto which that vertex can be placed. Obviously, for fixed vertices, $I_v = f(v)$. For free vertices, let I'_v be the set of positions that have at least $l(v)$ positions before them and at least $h(v)$ positions after them. Let $m_v \coloneqq \min I'_v$. Then $I_v = \{i \in I'_v : \forall u \in Dom(d_v) : m_u + d(u,v) \leq i\}$, where $Dom(d_v) \coloneqq \{u \in V : d(u,v) \text{ is defined}\}$.

The algorithm is as follows:

1. Let $\pi(v) = i$, where $f(v) = \{i\}$ for all fixed vertices v.
2. Go through the free positions in increasing order. Let i be the current position:
   (a) Let v be a vertex for which $i \in I_v$ and $\max I_v$ is minimal.
   (b) If no such v exists, then there is no solution.
   (c) If such a v exists, then let $\pi(v) = i$, delete i from every I_u, and let $I_v \coloneqq \emptyset$.

If the algorithm assigns a free position to every free vertex, then the problem has a solution, and the function π specifies an appropriate topological ordering.

3.3 NP-hardness

Although an efficient greedy solution exists for the task if a single position or an interval of positions is prescribed for certain vertices, we demonstrated that in some cases the problem is NP-hard.

Consider the variant of the problem where the graph G is a union of directed stars, and two or three positions are prescribed for every vertex (meaning $|f(v)| \in \{2,3\}$ for every vertex $v \in V$). In this case, deciding whether a position-specific ordering exists is an NP-hard problem. We will prove the NP-hardness by reduction from the 3-SAT problem.

Consider an arbitrary 3-SAT problem instance containing the literals $x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n$ and the clauses $c_1, c_2, \ldots, c_m$. Let the three literals appearing in clause c_j be c_j^1, c_j^2, c_j^3 for every $j \in \{1, \ldots, m\}$, which we shall call the *representatives* of the clause. Furthermore, construct a graph G whose vertices are the literals and the clause representatives, where a directed edge xc goes from a literal x to a representative c if the two are identical (no directed edge leaves any representative to any literal). Thus, the graph G is a union of directed stars and has a total of $2n+3m$ vertices. Let $f(x_i) = f(\overline{x}_i) = \{i, 2n+m-i+1\}$ for each $1 \leq i \leq n$, and let $f(c_j^1) = f(c_j^2) = f(c_j^3) = \{n+j, 2n+3m-2j+1, 2n+3m-2j+2\}$ for each $1 \leq j \leq m$. We will prove that the 3-SAT problem has a solution if and only if there is a topological ordering for the graph G with respect to the function f.

First, suppose that a solution to the 3-SAT instance exists. Then, place the vertices of graph G as follows:

1. For each $1 \leq i \leq n$, out of the literals x_i and $\overline{x}_i$, the one that takes a true value in the 3-SAT solution should be placed in position i. The other one should be placed in position $2n + m - i + 1$.

2. For each $1 \leq j \leq m$, the representative of clause c_j that is true in the solution should be placed in position $n + j$. If there are multiple such representatives, any of them can be chosen. The other two representatives should be placed in positions $2n + 3m - 2j + 1$ and $2n + 3m - 2j + 2$.

It can be seen that this sequence satisfies the position constraints imposed by f, so we only need to show that the constructed sequence is indeed a topological ordering.

Since every arc goes from a literal to a representative, and the first n elements of the sequence are literals, the next m elements are representatives, and the subsequent n elements are literals again, it is sufficient to show that there is no arc leading from the second set of n literals to the first set of m representatives. This follows immediately because, by construction, the first m representatives are all true, while the second set of n literals are all false.

Now, suppose that a topological ordering of graph G exists with function f. Take such an ordering. In this sequence, similarly, the first n elements are literals, the following m elements are representatives, and the next n elements are literals again. Assign truth value true to the first n literals. We will show that every clause becomes true under this assignment. Consider the first m representatives. Since each representative has (exactly one) incoming arc, due to the topological ordering, it can only come from one of the first n literals; therefore, the clause containing the given representative will be true. Since each clause has exactly one representative among the first m representatives, every clause will be true, meaning that a solution to the 3-SAT problem exists.

We have thus shown that if G is a union of directed stars and two or three positions are prescribed for each vertex, then solving the problem is NP-hard. We attempted to extend the above construction to other constraints as well. First, we modified the prescriptions such that the union of two disjoint intervals was assigned to each vertex in the following way: for every $1 \leq i \leq n$, let $f'(x_i) = f'(\overline{x}_i) = [1, i] \cup [2n + m - i + 1, 2n + m]$, and for every $1 \leq j \leq m$, let $f'(c_j^1) = f'(c_j^2) = f'(c_j^3) = [1, n + j] \cup [2n + 3m - 2j + 1, 2n + 3m]$. We will prove that solving the problem is NP-hard in this case as well.

If we assume that the 3-SAT problem has a solution, the construction seen in the previous proof remains valid here, so this direction directly follows from the previous results. For the other direction, we will show that a topological ordering satisfying the conditions of f' always satisfies the conditions of the previous function f as well.

Take a topological ordering that satisfies f'. Then, literals can only be placed into the first n positions and the interval $[n + m + 1, 2n + m]$. There are exactly $2n$ such elements, which means that only literals can occupy these positions. Consequently, only representatives can occupy the positions in $[n + 1, n + m]$ and $[2n + m + 1, 2n + 3m]$.

Consider the literals x_1 and $\overline{x}_1$. These literals can only be placed in positions 1 and $2n + m$; however, in this case, the literals x_2 and $\overline{x}_2$ can only be placed in positions 2 and $2n + m - 1$. Continuing this line of reasoning, it is easy to see that x_i and $\overline{x}_i$ can only be placed in positions i and $2n + m - i + 1$, which complies with the prescriptions of f. Similarly, it can be shown that the positions of the representatives also satisfy the prescriptions of f, meaning the previous result can be applied.

4 Summary and future plans

During the semester, we extended the concept of spectrum to two vertices and formulated conjectures for determining the common spectrum of three or more vertices in a directed tree; furthermore, we examined the problem of the existence of a position-specific ordering, which is closely related to the problem of determining spectra.

The goal for the next semester will be to verify our conjecture regarding spectra and to investigate other variants of position-specific ordering problems.

References

[1] M. D. Atkinson. On computing the number of linear extensions of a tree. *Order*, 7(1):23–25, 1990.

[2] G. Brightwell and P. Winkler. Counting linear extensions is #P-complete. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, pages 175–181, 1991.

[3] T. Hagerup and M. Maas. Generalized topological sorting in linear time. *Nord. J. Comput.*, 1(1):38–49, 1994.

[4] K. Kangas, T. Hankala, T. M. Niinimäki, and M. Koivisto. Counting linear extensions of sparse posets. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 603–609, 2016.

[5] K. Kangas, M. Koivisto, and S. Salonen. A faster tree-decomposition based algorithm for counting linear extensions. *Algorithmica*, 82(8):2156–2173, 2020.