Typical Sequences Revisited — Computing Width Parameters of Graphs*

Hans L. Bodlaender[1] Lars Jaffke[2] Jan Arne Telle[2]

[1]Utrecht University, The Netherlands
h.l.bodlaender@uu.nl
[2]University of Bergen, Norway
{lars.jaffke,jan.arne.telle}@uib.no

January 13, 2020

Abstract

In this work, we give a structural lemma on merges of typical sequences, a notion that was introduced in 1991 [Lagergren and Arnborg, Bodlaender and Kloks, both ICALP 1991] to obtain constructive linear time parameterized algorithms for treewidth and pathwidth. The lemma addresses a runtime bottleneck in those algorithms but so far it does not lead to asymptotically faster algorithms. However, we apply the lemma to show that the cutwidth and the modified cutwidth of series parallel digraphs can be computed in $\mathcal{O}(n^2)$ time.

1 Introduction

In this paper we revisit an old key technique from what currently are the theoretically fastest parameterized algorithms for treewidth and pathwidth, namely the use of *typical sequences*, and give additional structural insights for this technique. In particular, we show a structural lemma, which we call the *Merge Dominator Lemma*. The technique of typical sequences brings with it a partial ordering on sequences of integers, and a notion of possible merges of two integer sequences; surprisingly, the Merge Dominator Lemma states that for any pair of integer sequences there exists a *single* merge that dominates all merges of these integer sequences, and this dominating merge can be found in linear time. On its own, this lemma does not lead to asymptotically faster parameterized algorithms for treewidth and pathwidth, but, as we discuss below, it is a concrete step towards such algorithms.

The notion of typical sequences was introduced independently in 1991 by Lagergren and Arnborg [15] and Bodlaender and Kloks [8]. In both papers, it is a key element in an explicit dynamic programming algorithm that given a tree decomposition of bounded width ℓ, decides if the pathwidth or treewidth of the input graph G is at most a constant k. Lagergren and Arnborg build upon this result and show that the set of forbidden minors of graphs of treewidth (or pathwidth) at most k is computable; Bodlaender and Kloks show that the algorithm can also construct a tree or path decomposition of width at most k, if existing, in the same asymptotic

*This work was started when the third author was visiting Universitat Politecnica de Valencia, and part of it was done while the second author was visiting Utrecht University. The first author was partially supported by the Networks project, funded by the Netherlands Organization for Scientific Research (NWO). The second author is supported by the Bergen Research Foundation (BFS).

time bounds. The latter result is a main subroutine in Bodlaender's linear time algorithm [3] for treewidth-k. If one analyses the running time of Bodlaender's algorithm for treewidth or pathwidth $\leq k$, then one can observe that the bottleneck is in the subroutine that calls the Bodlaender-Kloks dynamic programming subroutine, with both the subroutine and the main algorithm having time $\mathcal{O}(2^{\mathcal{O}(k^3)}n)$ for treewidth, and $\mathcal{O}(2^{\mathcal{O}(k^2)}n)$ for pathwidth. See also the recent work by Fürer for pathwidth [13], and the simplified versions of the algorithms of [3, 8] by Althaus and Ziegler [1]. Now, over a quarter of a century after the discovery of these results, even though much work has been done on treewidth recognition algorithms (see e.g. [2, 5, 11, 12, 13, 14, 16, 17]), these bounds on the function of k are still the best known, i.e. no $\mathcal{O}(2^{o(k^3)}n^{O(1)})$ algorithm for treewidth, and no $\mathcal{O}(2^{o(k^2)}n^{O(1)})$ algorithm for pathwidth is known. An interesting question, and a long-standing open problem in the field [4, Problem 2.7.1], is whether such algorithms can be obtained. Possible approaches to answer such a question is to design (e.g. ETH or SETH based) lower bounds, find an entirely new approach to compute treewidth or pathwidth in a parameterized setting, or improve upon the dynamic programming algorithms of [15] and [8]. Using our Merge Dominator Lemma we can go one step towards the latter, as follows.

The algorithms of Lagergren and Arnborg [15] and Bodlaender and Kloks [8] are based upon tabulating characteristics of tree or path decompositions of subgraphs of the input graph; a characteristic consists of an *intersection model*, that tells how the vertices in the current top bag interact, and for each *part* of the intersection model, a typical sequence of bag sizes.[1] The set of characteristics for a join node is computed from the sets of characteristics of its (two) children. In particular, each pair of characteristics with one from each child can give rise to exponentially (in k) many characteristics for the join node. This is because exponentially many typical sequences may arise as the merges of the typical sequences that are part of the characteristics. In the light of our Merge Dominator Lemma, only *one* of these merges has to be stored, reducing the number of characteristics arising from each pair of characteristics of the children from $2^{\mathcal{O}(k)}$ to just 1. Moreover, this dominating merge can be found in $\mathcal{O}(k)$ time, with no large constants hidden in the '$\mathcal{O}$'.

Merging typical sequences at a join node is however not the only way the number of characteristics can increase throughout the algorithm, e.g. at introduce nodes, the number of characteristics increases in a different way. Nevertheless, the number of intersection models is $\mathcal{O}(k^{\mathcal{O}(k)})$ for pathwidth and $\mathcal{O}(k^{\mathcal{O}(k^2)})$ for treewidth; perhaps, with additional techniques, the number of typical sequences per part can be better bounded — in the case that a single dominating typical sequence per part suffices, this would reduce the number of table entries per node to $\mathcal{O}(k^{\mathcal{O}(k)})$ for pathwidth-k, and to $\mathcal{O}(k^{\mathcal{O}(k^2)})$ for treewidth-k, and yield $\mathcal{O}(k^{\mathcal{O}(k)}n)$ and $\mathcal{O}(k^{\mathcal{O}(k^2)}n)$ time algorithms for the respective problems.

We give direct algorithmic consequences of the Merge Dominator Lemma in the realm of computing width parameters of directed acyclic graphs (DAGs). Specifically, we show that the (Weighted) Cutwidth and Modified Cutwidth problems on DAGs, which given a directed acyclic graph on n vertices, ask for the topological order that minimizes the *cutwidth* and *modified cutwidth*, respectively, can be solved in $\mathcal{O}(n^2)$ time on *series parallel digraphs*. Note that the restriction of the solution to be a *topological* order has been made as well in other works, e.g. [6].

Our algorithm for Cutwidth of series parallel digraphs has the same structure as the dynamic programming algorithm for undirected Cutwidth [6], but, in addition to obeying directions of edges, we have a step that only keeps characteristics that are not dominated by

[1]This approach was later used in several follow up results to obtain explicit constructive parameterized algorithms for other graph width measures, like cutwidth [18, 19], branchwidth [9], different types of search numbers like linear width [10], and directed vertex separation number [7].

another characteristic in a table of characteristics. Now, with help of our Merge Dominator Lemma, we can show that in the case of series parallel digraphs, there is a unique dominating characteristic; the dynamic programming algorithm reverts to computing for each intermediate graph a single 'optimal partial solution'. This strategy also works in the presence of edge weights, which gives the algorithm for the corresponding WEIGHTED CUTWIDTH problem on series parallel digraphs. Note that the cutwidth of a directed acyclic graph is at least the maximum indegree or outdegree of a vertex; e.g., a series parallel digraph formed by the parallel composition of $n-2$ paths with three vertices has n vertices and cutwidth $n-2$. To compute the *modified* cutwidth of a series parallel digraph, we give a linear-time reduction to the WEIGHTED CUTWIDTH problem on series parallel digraphs.

This paper is organized as follows. In Section 2, we give a number of preliminary definitions, and review existing results, including several results on typical sequences from [8]. In Section 3, we state and prove the main technical result of this work, the Merge Dominator Lemma. Section 4 gives our algorithmic applications of this lemma, and shows that the directed cutwidth and directed modified cutwidth of a series parallel digraph can be computed in polynomial time. Some final remarks are made in the conclusions Section 5.

2 Preliminaries

We use the following notation. For two integers $a, b \in \mathbb{N}$ with $a \leq b$, we let $[a..b] := \{a, a+1, \ldots, b\}$ and for $a > 0$, we let $[a] := [1..a]$. If X is a set of size n, then a *linear order* is a bijection $\pi \colon X \to [n]$. Given a subset $X' \subseteq X$ of size $n' \leq n$, we define the *restriction of* π *to* X' as the bijection $\pi|_{X'} \colon X' \to [n']$ which is such that for all $x', y' \in X'$, $\pi|_{X'}(x') < \pi|_{X'}(y')$ if and only if $\pi(x') < \pi(y')$.

Sequences and Matrices. We denote the elements of a sequence s by $s(1), \ldots, s(n)$. We denote the *length* of s by $l(s)$, i.e. $l(s) := n$. For two sequences $a = a(1), \ldots, a(m)$ and $b = b(1), \ldots, b(n)$, we denote their *concatenation* by $a \circ b = a(1), \ldots, a(m), b(1), \ldots, b(n)$. For two sets of sequences A and B, we let $A \odot B := \{a \circ b \mid a \in A \wedge b \in B\}$. For a sequence s of length n and a set $X \subseteq [n]$, we denote by $s[X]$ the *subsequence of* s *induced by* X, i.e. let $X = \{x_1, \ldots, x_m\}$ be such that for all $i \in [m-1]$, $x_i < x_{i+1}$; then, $s[X] := s(x_1), \ldots, s(x_m)$. For $x_1, x_2 \in [n]$ with $x_1 \leq x_2$, we use the shorthand '$s[x_1..x_2]$' for '$s[[x_1..x_2]]$'.

Let Ω be a set. A *matrix* $M \in \Omega^{m \times n}$ over Ω is said to have m rows and n columns. For sets $X \subseteq [m]$ and $Y \subseteq [n]$, we denote by $M[X, Y]$ the *submatrix* of M *induced* by X and Y, which consists of all the entries from M whose indices are in $X \times Y$. For $x_1, x_2 \in [m]$ with $x_1 \leq x_2$ and $y_1, y_2 \in [n]$ with $y_1 \leq y_2$, we use the shorthand '$M[x_1..x_2, y_1..y_2]$' for ''$M[[x_1..x_2], [y_1..y_2]]$'. For a sequence $s(1), s(2), \ldots, s(\ell)$ of indices of a matrix M, we let

$$M[s] := M[s(1)], M[s(2)], \ldots, M[s(\ell)] \tag{1}$$

be the corresponding sequence of entries from M.

For illustrative purposes we enumerate the columns of a matrix in a bottom-up fashion throughout this paper, i.e. we consider the index $(1,1)$ as the 'bottom left corner' and the index (m, n) as the 'top right corner'.

Integer Sequences. Let s be an integer sequence of length n. We use the shorthand '$\min(s)$' for '$\min_{i \in [n]} s(i)$' and '$\max(s)$' for '$\max_{i \in [n]} s(i)$'; we use the following definitions. We let

$$\operatorname{argmin}(s) := \{i \in [n] \mid s(i) = \min(s)\} \text{ and } \operatorname{argmax}(s) := \{i \in [n] \mid s(i) = \max(s)\}$$

be the set of indices at whose positions there are the minimum and maximum element of s, respectively. Whenever we write $i \in \operatorname{argmin}(s)$ ($j \in \operatorname{argmax}(s)$), then the choice of i (j) can be arbitrary. In some places we require a canonical choice of the position of a minimum or maximum element, in which case we will always choose the smallest index. Formally, we let

$$\operatorname{argmin}^\star(s) := \min \operatorname{argmin}(s), \text{ and } \operatorname{argmax}^\star(s) := \min \operatorname{argmax}(s).$$

The following definition contains two notions on pairs of integer sequences that are necessary for the definitions of domination and merges.

Definition 2.1. Let r and s be two integer sequences of the same length n.

(i) If for all $i \in [n]$, $r(i) \leq s(i)$, then we write '$r \leq s$'.

(ii) We write $q = r + s$ for the integer sequence $q(1), \ldots, q(n)$ with $q(i) = r(i) + s(i)$ for all $i \in [n]$.

Definition 2.2 (Extensions). Let s be a sequence of length n. We define the set $E(s)$ of *extensions* of s as the set of sequences that are obtained from s by repeating each of its elements an arbitrary number of times, and at least once. Formally, we let

$$E(s) := \{s_1 \circ s_2 \circ \cdots \circ s_n \mid \forall i \in [n]\colon l(s_i) \geq 1 \wedge \forall j \in [l(s_i)]\colon s_i(j) = s(i)\}.$$

Definition 2.3 (Domination). Let r and s be integer sequences. We say that r *dominates* s, in symbols '$r \prec s$', if there are extensions $r^* \in E(r)$ and $s^* \in E(s)$ of the same length such that $r^* \leq s^*$. If $r \prec s$ and $s \prec r$, then we say that r and s are *equivalent*, and we write $r \equiv s$.

If r is an integer sequence and S is a set of integer sequences, then we say that r *dominates* S, in symbols '$r \prec S$', if for all $s \in S$, $r \prec s$.

Remark 2.4 (Transitivity of '$\prec$'). In [8, Lemma 3.7], it is shown that the relation '$\prec$' is transitive. As this is fairly intuitive, we may use this fact without stating it explicitly throughout this text.

Definition 2.5 (Merges). Let r and s be two integer sequences. We define the set of all *merges* of r and s, denoted by $r \oplus s$, as $r \oplus s := \{r^* + s^* \mid r^* \in E(r), s^* \in E(s), l(r^*) = l(s^*)\}$.

2.1 Typical Sequences

We now define typical sequences, show how to construct them in linear time, and restate several lemmas due to Bodlaender and Kloks [8] that will be used throughout this text.

Definition 2.6. Let $s = s(1), \ldots, s(n)$ be an integer sequence of length n. The *typical sequence of* s, denoted by $\tau(s)$, is obtained from s by an exhaustive application of the following two operations:

Removal of Consecutive Repetitions. If there is an index $i \in [n-1]$ such that $s(i) = s(i+1)$, then we change the sequence s from $s(1), \ldots, s(i), s(i+1), \ldots, s(n)$ to $s(1), \ldots, s(i), s(i+2), \ldots, s(n)$.

Typical Operation. If there exist $i, j \in [n]$ such that $j - i \geq 2$ and for all $i \leq k \leq j$, $s(i) \leq s(k) \leq s(j)$, or for all $i \leq k \leq j$, $s(i) \geq s(k) \geq s(j)$, then we change the sequence s from $s(1), \ldots, s(i), s(i+1), \ldots, s(j), \ldots, s(n)$ to $s(1), \ldots, s(i), s(j), \ldots, s(n)$, i.e. we remove all elements (strictly) between index i and j.

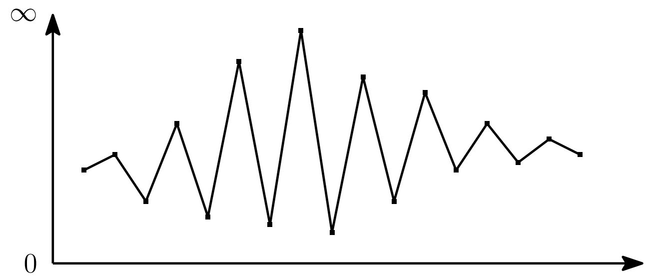


Figure 1: Illustration of the shape of a typical sequence.

To support intuition, we illustrate the rough shape of a typical sequence in Figure 1. It is not difficult to see that the typical sequence can be computed in quadratic time, by an exhaustive application of the definition. Here we discuss how to do it in linear time. We may view a typical sequence $\tau(s)$ of an integer sequence s as a subsequence of s. While $\tau(s)$ is unique, the choice of indices that induce $\tau(s)$ may not be unique. We show that we can find a set of indices that induce the typical sequence in linear time, with help of the following structural proposition.

Proposition 2.7. *Let s be an integer sequence and let $i^\star \in \{\operatorname{argmin}^\star(s), \operatorname{argmax}^\star(s)\}$. Let $1 =: j_0 < j_1 < j_2 < \ldots < j_t < j_{t+1} := i^\star$ be pairwise distinct integers, such that $s(j_0), \ldots, s(j_{t+1})$ are pairwise distinct. If for all $h \in [0..t]$,*

- *if $s(j_h) > s(j_{h+1})$ then $j_h = \operatorname{argmax}^\star(s[1..j_{h+1}])$ and $j_{h+1} = \operatorname{argmin}^\star(s[1..j_{h+1}])$, and*
- *if $s(j_h) < s(j_{h+1})$ then $j_h = \operatorname{argmin}^\star(s[1..j_{h+1}])$ and $j_{h+1} = \operatorname{argmax}^\star(s[1..j_{h+1}])$,*

then the typical sequence of s restricted to $[i^\star]$ is equal to $s(j_0), s(j_1), \ldots, s(j_t), s(j_{t+1})$.

Proof. First, we observe that by the choice made in the definition of argmin* and argmax*,

$$\text{for each } h \in [0..(t+1)] \text{ there is no } i < j_h \text{ such that } s(i) = s(j_h). \tag{2}$$

We prove the following statement. Under the stated conditions, for a given $h \in [0..t+1]$, the typical sequence of s restricted to $[j_h..i^\star]$ is equal to $s(j_h)$, $s(j_{h+1})$, $\ldots$, $s(j_{t+1})$. The proposition then follows from the case $h = 0$. The proof is by induction on $d := (t+1) - h$. For $d = 0$, it trivially holds since the minimum and the maximum element are always part of the typical sequence, and since $[j_{t+1}..i^\star] = \{i^\star\}$.

Now suppose $d > 0$, and for the induction hypothesis, that the claim holds for $d-1$. Suppose that $s(j_h) > s(j_{h+1})$, meaning that $j_h = \operatorname{argmax}^\star(s[1..j_{h+1}])$, and $j_{h+1} = \operatorname{argmin}^\star(s[1..j_{h+1}])$, the other case is symmetric. By the induction hypothesis, the typical sequence of s restricted to $[j_{h+1}..i^\star]$ is equal to $s(j_{h+1})$, $\ldots$, $s(j_{t+1})$, in particular it implies that $s(j_{h+1})$ is an element of the typical sequence. To prove the induction step, we have to show that the typical sequence restricted to $[j_h..j_{h+1}]$ is equal to $s(j_h)$, $s(j_{h+1})$. We first argue that if there is an element of the typical sequence in $[j_h..(j_{h+1}-1)]$, then it must be equal to $s(j_h)$. By (2), we have that there is no $i < j_{h+1}$ such that $s(i) = s(j_{h+1})$, hence $[j_h..(j_{h+1}-1)]$ cannot contain any element of the typical sequence that is equal to $s(j_{h+1})$. Next, since the typical operation removes all elements $i \in [(j_h+1)..(j_{h+1}-1)]$ with $s(j_h) > s(i) > s(j_{h+1})$, and since $j_h = \operatorname{argmax}^\star(s[1..j_{h+1}])$, the only elements from $[j_h..(j_{h+1}-1)]$ that the typical sequence may contain have value $s(j_h)$.

It remains to argue that $s(j_h)$ is indeed an element of the typical sequence. Suppose not, then there are indices i, i' with $i < j_h < i'$, such that either $s(i) \le s(j_h) \le s(i')$, or $s(i) \ge s(j_h) \ge s(i')$, and we may assume that at least one of the inequalities is strict in each case. For the latter case, since $j_h = \operatorname{argmax}^\star(s[1..j_{h+1}])$, we would have that $s(i) = s(j_h)$, which is a contradiction to (2). Hence, we may assume that $s(i) \leq s(j_h) \leq s(i')$. There are two cases to consider: $i' \in [(j_h+1)..j_{h+1}]$, and $i' > j_{h+1}$. If $i' \in [(j_h+1)..j_{h+1}]$, then $s(i') = s(j_h)$, as $s(j_h) = \text{argmax}(s[1..j_{h+1}])$. We can conclude that in this case, the typical sequence must contain an element equal to $s(i')$, and hence equal to $s(j_h)$. If $i' > j_{h+1}$, then the typical operation corresponding to i and i' also removes $s(j_{h+1})$, a contradiction with the induction hypothesis which asserts that $s(j_{h+1})$ is part of the typical sequence induced by $[j_{h+1}..i^\star]$. We can conclude that $s(j_h)$ is part of the typical sequence, finishing the proof. □

From the previous proposition, we have the following consequence about the structure of typical sequences ending in the minimum element, which will be useful in the proof of Lemma 3.10.

Corollary 2.8. Let t be a typical sequence of length n such that $n \in \text{argmin}(t)$. Then, for each $k \in \left[\lfloor \frac{n}{2} \rfloor\right]$, $n-2k+1 \in \text{argmax}(t[1..(n-2k+1)])$ and $n-2k \in \text{argmin}(t[1..(n-2k)])$.

Equipped with Proposition 2.7, we can now proceed and give the linear-time algorithm that computes a typical sequence of an integer sequence.

Lemma 2.9. *Let s be an integer sequence of length n. Then, one can compute $\tau(s)$, the typical sequence of s, in time $\mathcal{O}(n)$.*

Proof. First, we check for each $i \in [n-1]$ whether $s(i) = s(i+1)$, and if we find such an index i, we remove $s(i)$. We assume from now on that after these modifications, s has at least two elements, otherwise it is trivial. As observed above, the typical sequence of s contains $\min(s)$ and $\max(s)$. A closer look reveals the following observation.

Observation 2.9.1. Let $i^\star := \min \text{argmin}(s) \cup \text{argmax}(s)$ and $k^\star := \max \text{argmin}(s) \cup \text{argmax}(s)$.

(i) If $i^\star \in \text{argmin}(s)$ and $k^\star \in \text{argmax}(s)$ or $i^\star \in \text{argmax}(s)$ and $k^\star \in \text{argmin}(s)$, then $\tau(s)$ restricted to $[i^\star..k^\star]$ is equal to $s(i^\star), s(k^\star)$.

(ii) If $\{i^\star, k^\star\} \subseteq \text{argmin}(s)$, then $\tau(s)$ restricted to $[i^\star..k^\star]$ is equal to $s(i^\star), \max(s), s(k^\star)$.

(iii) If $\{i^\star, k^\star\} \subseteq \text{argmax}(s)$, then $\tau(s)$ restricted to $[i^\star..k^\star]$ is equal to $s(i^\star), \min(s), s(k^\star)$.

Let $i^\star := \min \text{argmin}(s) \cup \text{argmax}(s)$ and $k^\star := \max \text{argmin}(s) \cup \text{argmax}(s)$. Using Observation 2.9.1, it remains to determine the indices that induce the typical sequence on $s[1..i^\star]$ and on $s[k^\star..n]$. To find the indices that induce the typical sequence on $s[1..i^\star]$, we will describe a marking procedure that marks a set of indices satisfying the preconditions of Proposition 2.7. Next, we observe that $n - k^\star$ is the *smallest* index of any occurrence of $\min(s)$ or $\max(s)$ in the *reverse* sequence of s, therefore a symmetric procedure, again using Proposition 2.7, yields the indices that induce $\tau(s)$ on $s[k^\star..n]$.

We execute Algorithm 1, which processes the integer sequence $s[1..i^\star]$ from the first to the last element, storing two counters $j_{\min}$ and $j_{\max}$ that store the leftmost position of the smallest and of the greatest element seen so far, respectively. Whenever a new minimum is encountered, we mark the current value of the index $j_{\max}$, as this implies that $s(j_{\max})$ has to be an element of the typical sequence. Similarly, when encountering a new maximum, we mark $j_{\min}$. These marked indices are stored in a set M, which at the end of the algorithm contains the indices that induce $\tau(s)$ on $[1..i^\star]$. This, i.e. the correctness of the procedure, will now be argued via Proposition 2.7.

Claim 2.9.2. The set M of indices marked by the above procedure induce $\tau(s)$ on $[1..i^\star]$.

$j_{\min} \leftarrow \text{argmin}^\star(s[1..2])$, $j_{\max} \leftarrow \text{argmax}^\star(s[1..2])$, $M \leftarrow \{1\}$
for $j = 3, \ldots, i^\star$ **do**
  if $s(j) < s(j_{\min})$ **then**
    $j_{\min} \leftarrow j$
    $M \leftarrow M \cup \{j_{\max}\}$ `// mark the current value of` $j_{\max}$
  if $s(j) > s(j_{\max})$ **then**
    $j_{\max} \leftarrow j$
    $M \leftarrow M \cup \{j_{\min}\}$ `// mark the current value of` $j_{\min}$
$M \leftarrow M \cup \{j_{\min}, j_{\max}\}$

Algorithm 1: The algorithm of Lemma 2.9 that computes the set M of indices that induce the typical sequence of s between the first element and the first occurrence of the minimum and maximum of s.

Proof. Let $M = \{j_0, j_1, \ldots, j_{t+1}\}$ be such that for all $h \in [0..t]$, $j_h < j_{h+1}$. We prove that $j_0, \ldots, j_{t+1}$ meet the preconditions of Proposition 2.7. First, we observe that the above algorithm marks both the index 1 and index $i^\star$, in particular that $j_0 = 1$ and $j_{t+1} = i^\star$.

We verify that the indices $j_0, \ldots, j_{t+1}$ satisfy the property that for each $[0..(t+1)]$, the index j_h is the leftmost (i.e. smallest) index whose value is equal to $s(j_h)$: whenever an index is added to the marked set, it is because in some iteration, the element at its position was either strictly greater than the greatest previously seen element, or strictly smaller than the smallest previously seen element. (This also ensures that $s(j_0), \ldots, s(j_{t+1})$ are pairwise distinct.)

We additionally observe that if we have two indices ℓ_1 and ℓ_2 such that ℓ_2 is the index that the algorithm marked right after it marked ℓ_1, then either ℓ_1 was $j_{\min}$ and ℓ_2 was $j_{\max}$ or vice versa: when updating $j_{\min}$, we mark $j_{\max}$, and when updating $j_{\max}$, we mark $j_{\min}$. This lets us conclude that when we have two indices j_h, j_{h+1} such that $s(j_h) < s(j_{h+1})$, then j_h was equal to $j_{\min}$ when it was marked, and j_{h+1} was $j_{\max}$ when it was marked.

We are ready to prove that $j_0, \ldots, j_{t+1}$ satisfy the precondition of Proposition 2.7. Suppose for a contradiction that for some $h \in [0..t+1]$, j_h violates this property. Assume that $s(j_h) < s(j_{h+1})$ and note that the other case is symmetric. The previous paragraph lets us conclude that j_h was equal to $j_{\min}$ when it was marked, and that j_{h+1} was $j_{\max}$ when it was marked.

We may assume that either $j_h \neq \text{argmin}^\star(s[1..j_{h+1}])$ or that $j_{h+1} \neq \text{argmax}^\star(s[1..j_{h+1}])$. Suppose the latter holds. This immediately implies that there is some $j^* \in [j_{h+1} - 1]$ such that $s(j^*) > j_{h+1}$, which implies that $j_{\max}$ would never have been set to j_{h+1} and hence j_{h+1} would have never been marked. Suppose the former holds, i.e. $j_h \neq \text{argmin}^\star(s[1..j_h])$, for an illustration of the following argument see Figure 2. Let $j^* := \text{argmin}^\star(s[1..j_{h+1}])$. If $j^* < j_h$, then at iteration j_h, $s(j_{\min}) < s(j_h)$, so $j_{\min}$ would never have been set to j_h, and hence, j_h would never have been marked. We may assume that $j^* > j_h$. Since j_h was marked, there is some $\ell > j_h$ that triggered j_h being marked. This also means that at that iteration $s(\ell)$ was greater than the previously observed maximum, so we may assume that $s(\ell) > s(j_h)$. We also may assume that $\ell \leq j_{h+1}$. If $j^* \in [(j_h + 1)..(\ell - 1)]$, then the algorithm would have updated $j_{\min}$ to j^* in that iteration, before marking j_h, and for the case $j^* \in [(\ell + 1)..(j_{h+1} - 1)]$ we observe that $\ell \neq j_{h+1}$, and that the algorithm would mark ℓ as the next index instead of j_{h+1}. ⌟

This establishes the correctness of the algorithm. For its runtime, we observe that each iteration takes $\mathcal{O}(1)$ time, and that there are $\mathcal{O}(n)$ iterations. □

We summarize several lemmas from [8] regarding integer sequences and typical sequences that we will use in this work.

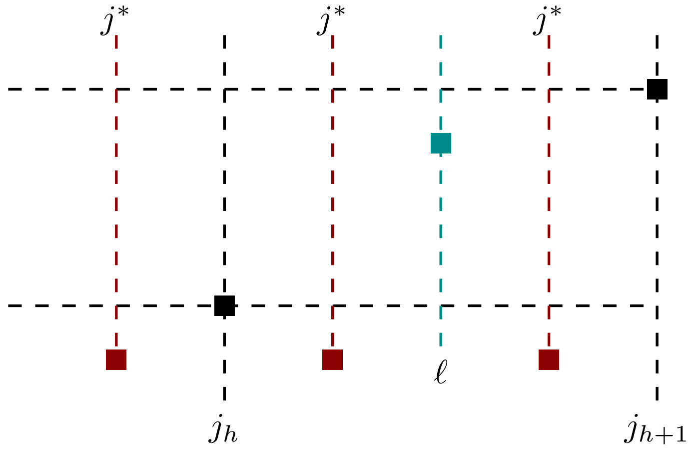


Figure 2: Illustration of the final argument in the proof of Claim 2.9.2. We assume that $s(j_h) < s(j_{h+1})$, and mark the possible positions for $j^* = \text{argmin}^\star(s[1..j_{h+1}])$ with $j^* \neq j_h$.

Lemma 2.10 (Bodlaender and Kloks [8]). *Let r and s be two integer sequences.*

(i) *(Cor. 3.11 in [8]). We have that $r \prec s$ if and only if $\tau(r) \prec \tau(s)$.*

(ii) *(Lem. 3.13 in [8]). Suppose r and s are of the same length and let $y = r + s$. Let $r_0 \prec r$ and $s_0 \prec s$. Then there is an integer sequence $y_0 \in r_0 \oplus s_0$ such that $y_0 \prec y$.*

(iii) *(Lem. 3.14 in [8]). Let $q \in r \oplus s$. Then, there is an integer sequence $q' \in \tau(r) \oplus \tau(s)$ such that $q' \prec q$.*

(iv) *(Lem. 3.15 in [8]). Let $q \in r \oplus s$. Then, there is an integer sequence $q' \in r \oplus s$ with $\tau(q') = \tau(q)$ and $l(q') \leq l(r) + l(s) - 1$.*

(v) *(Lem. 3.19 in [8]). Let r' and s' be two more integer sequences. If $r' \prec r$ and $s' \prec s$, then $r' \circ s' \prec r \circ s$.*

2.2 Directed Acyclic Graphs

A *directed graph* (or *digraph*) G is a pair of a set of *vertices* $V(G)$ and a set of ordered pairs of vertices, called *arcs*, $A(G) \subseteq V(G) \times V(G)$. (If $A(G)$ is a multiset, we call G *multidigraph*.) We say that an arc $a = (u, v) \in A(G)$ is directed from u to v, and we call u the *tail* of a and v the *head* of a. We use the shorthand 'uv' for '(u, v)'. A sequence of vertices $v_1, \ldots, v_r$ is called a *walk* in G if for all $i \in [r-1]$, $v_i v_{i+1} \in A(G)$. A *cycle* is a walk $v_1, \ldots, v_r$ with $v_1 = v_r$ and all vertices $v_1, \ldots, v_{r-1}$ pairwise distinct. If G does not contain any cycles, then we call G *acyclic* or a *directed acyclic graph*, DAG for short.

Let G be a DAG on n vertices. A *topological order* of G is a linear order $\pi \colon V(G) \to [n]$ such that for all arcs $uv \in A(G)$, we have that $\pi(u) < \pi(v)$. We denote the set of all topological orders of G by $\Pi(G)$. We now define the width measures studied in this work. Note that we restrict the orderings of the vertices that we consider to *topological* orderings.

Definition 2.11. Let G be a directed acyclic graph and let $\pi \in \Pi(G)$ be a topological order of G.

(i) The *cutwidth* of π is $\mathsf{cutw}(\pi) := \max_{i \in [n-1]} |\{uv \in A(G) \mid \pi(u) \leq i \wedge \pi(v) > i\}|$.

(ii) The *modified cutwidth* of π is $\mathsf{mcutw}(\pi) := \max_{i \in [n]} |\{uv \in A(G) \mid \pi(u) < i \wedge \pi(v) > i\}|$.

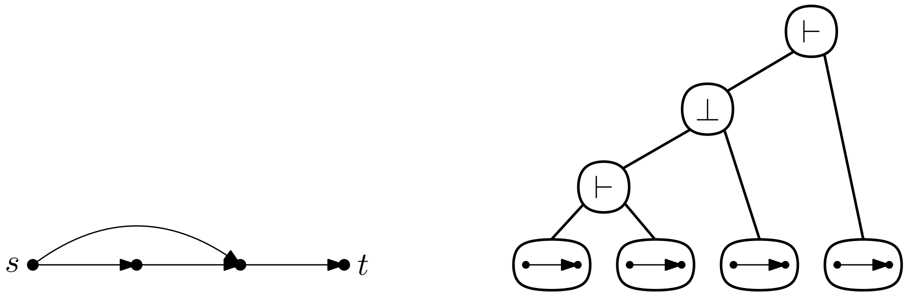


Figure 3: A series parallel digraph G on the left, and a decomposition tree that yields G on the right.

We define the cutwidth and modified cutwidth of a directed acyclic graph G as the minimum of the respective measure over all topological orders of G.

We now introduce series parallel digraphs. Note that the following definition coincides with the notion of 'edge series-parallel multidigraphs' in [20]. For an illustration see Figure 3.

Definition 2.12 (Series Parallel Digraph (SPD)). A (multi-)digraph G with an ordered pair of *terminals* $(s, t) \in V(G) \times V(G)$ is called *series parallel digraph (SPD)*, often denoted by $(G, (s, t))$, if one of the following hold.

(i) $(G, (s, t))$ is a single arc directed from s to t, i.e. $V(G) = \{s, t\}$, $A(G) = \{(s, t)\}$.

(ii) $(G, (s, t))$ can be obtained from two series parallel digraphs $(G_1, (s_1, t_1))$ and $(G_2, (s_2, t_2))$ by one of the following operations.

   (a) *Series Composition.* $(G, (s, t))$ is obtained by taking the disjoint union of G_1 and G_2, identifying t_1 and s_2, and letting $s = s_1$ and $t = t_2$. In this case we write $(G, (s, t)) = (G_1, (s_1, t_1)) \vdash (G_2, (s_2, t_2))$ or simply $G = G_1 \vdash G_2$.

   (b) *Parallel Composition.* $(G, (s, t))$ is obtained by taking the disjoint union of G_1 and G_2, identifying s_1 and s_2, and identifying t_1 and t_2, and letting $s = s_1 = s_2$ and $t = t_1 = t_2$. In this case we write $(G, (s, t)) = (G_1, (s_1, t_1)) \perp (G_2, (s_2, t_2))$, or simply $G = G_1 \perp G_2$.

It is not difficult to see that each series parallel digraph is acyclic. One can naturally associate a notion of *decomposition trees* with series parallel digraphs as follows. A decomposition tree T is a rooted and ordered binary tree whose leaves are labeled with a single arc, and each internal node $t \in V(T)$ with left child ℓ and right child r is either a *series node* or a *parallel node*. We then associate an SPD G_t with t that is $G_\ell \vdash G_r$ if t is a series node and $G_\ell \perp G_r$ if t is a parallel node. It is clear that for each SPD G, there is a decomposition tree T with root $\mathfrak{r}$ such that $G = G_{\mathfrak{r}}$. In that case we say that T *yields* G. Valdes et al. [20] have shown that one can decide in linear time whether a directed graph G is an SPD and if so, find a decomposition tree that yields G.

Theorem 2.13 (Valdes et al. [20]). *Let G be a directed graph on n vertices and m arcs. There is an algorithm that decides in time $\mathcal{O}(n + m)$ whether G is a series parallel digraph and if so, it outputs a decomposition tree that yields G.*

3 The Merge Dominator Lemma

In this section we prove the main technical result of this work. It states that given two integer sequences, one can find in linear time a merge that dominates all merges of those two sequences.

Lemma 3.1 (Merge Dominator Lemma). *Let r and c be integer sequence of length m and n, respectively. There exists a dominating merge of r and c, i.e. an integer sequence $t \in r \oplus c$ such that $t \prec r \oplus c$, and this dominating merge can be computed in time $\mathcal{O}(m + n)$.*

Outline of the proof of the Merge Dominator Lemma. First, we show that we can restrict our search to finding a dominating path in a matrix that, roughly speaking, contains all merges of r and c of length at most $l(r) + l(c) - 1$. The goal of this step is mainly to increase the intuitive insight to the proofs in this section. Next, we prove the 'Split Lemma' (Lemma 3.7 in Subsection 3.2) which asserts that we can obtain a dominating path in our matrix M by splitting M into a submatrix M_1 that lies in the 'bottom left' of M and another submatrix M_2 in the 'top right' of M along a minimum row and a minimum column, and appending a dominating path in M_2 to a dominating path in M_1. In M_1, the last row and column are a minimum row and column, respectively, and in M_2, the first row and column are a minimum row and column, respectively. This additional structure will be exploited in Subsection 3.3 where we prove the 'Chop Lemmas' that come in two versions. The 'bottom version' (Lemma 3.10) shows that in M_1, we can find a dominating path by repeatedly chopping away the *last* two rows or columns and remembering a vertical or horizontal length-2 path. The 'top version' (Corollary 3.12) is the symmetric counterpart for M_2. The proofs of the Chop Lemmas only hold when r and c are *typical sequences*, and in Subsection 3.4 we present the 'Split-and-Chop Algorithm' that computes a dominating path in a merge matrix of two typical sequences. Finally, in Subsection 3.5, we generalize this result to arbitrary integer sequences, using the Split-and-Chop Algorithm and one additional construction.

3.1 The Merge Matrix, Paths, and Non-Diagonality

Let us begin by defining the basic notions of a merge matrix and paths in matrices.

Definition 3.2 (Merge Matrix). Let r and c be two integer sequences of length m and n, respectively. Then, the *merge matrix* of r and c is an $m \times n$ integer matrix M such that for $(i, j) \in [m] \times [n]$, $M[i, j] = r(i) + c(j)$.

Definition 3.3 (Path in a Matrix). Let M be an $m \times n$ matrix. A *path* in M is a sequence $p(1), \ldots, p(\ell)$ of indices from M such that

(i) $p(1) = (1, 1)$ and $p(\ell) = (m, n)$, and

(ii) for $h \in [\ell - 1]$, let $p(h) = (i, j)$; then, $p(h + 1) \in \{(i + 1, j), (i, j + 1), (i + 1, j + 1)\}$.

We denote by $\mathcal{P}(M)$ the set of all paths in M. A sequence $p(1), \ldots, p(\ell)$ that satisfies the second condition but not necessarily the first is called a *partial path* in M. For two paths $p, q \in \mathcal{P}(M)$, we may simply say that p *dominates* q, if $M[p]$ dominates $M[q]$.[2] We also write $p \prec \mathcal{P}(M)$ to express that for each path $q \in \mathcal{P}(M)$, $p \prec q$.

A (partial) path is called *non-diagonal* if the second condition is replaced by the following.

(ii)' For $h \in [\ell - 1]$, let $p(h) = (i, j)$; then, $p(h + 1) \in \{(i + 1, j), (i, j + 1)\}$.

An *extension* e of a path p in a matrix M is as well a sequence of indices of M, and we again denote the corresponding integer sequence by $M[e]$. A consequence of Lemma 2.10(i) and (iv) is that we can restrict ourselves to all paths in a merge matrix when trying to find a dominating merge of two integer sequences: it is clear from the definitions that in a merge

[2] Recall that by (1) on page 3, for a (partial) path p in a matrix M, $M[p] = M[p(1)], M[p(2)], \ldots, M[p(l(p))]$.

matrix M of integer sequences r and c, $\mathcal{P}(M)$ contains all merges of r and c of length at most $l(r) + l(c) - 1$. Furthermore, suppose that there is a merge $q \in r \oplus s$ such that $q \prec r \oplus s$ and $l(q) > l(r)+l(s)-1$. By Lemma 2.10(iv), there is a merge $q' \in r \oplus s$ such that $l(q') \leq l(r)+l(s)-1$, and $\tau(q') = \tau(q)$. The latter yields $\tau(q') \equiv \tau(q)$ and therefore, by Lemma 2.10(i), $q' \equiv q$, in particular, $q' \prec q \prec r \oplus s$.

Corollary 3.4. Let r and c be integer sequences and M be the merge matrix of r and c. There is a dominating merge in $r \oplus c$, i.e. an integer sequence $t \in r \oplus c$ such that $t \prec r \oplus c$, if and only if there is a dominating path in M, i.e. a path $p \in \mathcal{P}(M)$ such that $p \prec \mathcal{P}(M)$.

We now consider a type of merge that corresponds to non-diagonal paths in the merge matrix. These merges will be used in a construction presented in Subsection 3.5, and in the algorithmic applications of the Merge Dominator Lemma given in Section 4. For two integer sequences r and s, we denote by $r \boxplus s$ the set of all *non-diagonal merges* of r and s, which are not allowed to have 'diagonal' steps: we have that for all $t \in r \boxplus s$ and all $i \in [l(t) - 1]$, if $t(i) = r(i_r) + s(i_s)$, then $t(i + 1) \in \{r(i_r + 1) + s(i_s), r(i_r) + s(i_s + 1)\}$. As each non-diagonal merge directly corresponds to a non-diagonal path in the merge matrix (and vice versa), we can consider a non-diagonal path in a merge matrix to be a non-diagonal merge and vice versa. We now show that for each merge that uses diagonal steps, there is always a non-diagonal merge that dominates it.

Lemma 3.5. *Let r and s be two integer sequences of length m and n, respectively. For any merge $q \in r \oplus s$, there is a non-diagonal merge $q' \in r \boxplus s$ such that $q' \prec q$. Furthermore, given q, q' can be found in time $\mathcal{O}(m + n)$.*

Proof. This can be shown by the following local observation. Let $i \in [l(q) - 1]$ be such that $q(i), q(i + 1)$ is a diagonal step, i.e. there are indices $i_r \in [l(r) - 1]$ and $i_s \in [l(s) - 1]$ such that $q(i) = r(i_r) + s(i_s)$ and $q(i + 1) = r(i_r + 1) + s(i_s + 1)$. Then, we insert the element $x := \min\{r(i_r) + s(i_s + 1), r(i_r + 1) + s(i_s)\}$ between $q(i)$ and $q(i + 1)$. Since

$$x \leq \max\{r(i_r) + s(i_s), r(i_r + 1), s(i_s + 1)\} =: y,$$

we can repeat y twice in an extension of q so that one of the occurrences aligns with x, and we have that in this position, the value of q' is at most the value of the extension of q.

Let q' be the sequence obtained from q by applying this operation to all diagonal steps, then by the observation just made, we have that $q' \prec q$. It is clear that this can be implemented to run in time $\mathcal{O}(m + n)$. $\square$

Next, we define two special paths in a matrix M that will reappear in several places throughout this section. These paths can be viewed as the 'corner paths', where the first one follows the first row until it hits the last column and then follows the last column ($p_\lrcorner(M)$), and the second one follows the first column until it hits the last row and then follows the last row ($p_\ulcorner(M)$). Formally, we define them as follows:

$$p_\lrcorner(M) := (1, 1), (1, 2), \dots, (1, n), (2, n), \dots, (m, n)$$
$$p_\ulcorner(M) := (1, 1), (2, 1) \dots, (m, 1), (m, 2), \dots, (m, n)$$

We use the shorthands '$p_\lrcorner$' for '$p_\lrcorner(M)$' and '$p_\ulcorner$' for '$p_\ulcorner(M)$' whenever M is clear from the context.

For instance, these paths appear in the following special cases of the Merge Dominator Lemma, which will be useful for several proofs in this section.

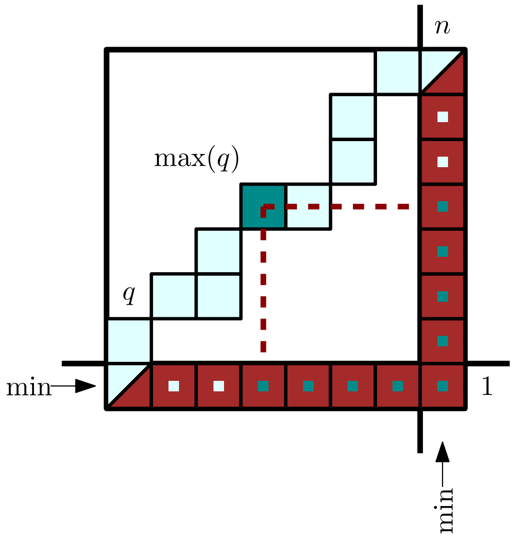


Figure 4: Situation in the proof of Lemma 3.6(i). The dot within each element of the corner path $p_{\lrcorner}$ indicates with which elements of the path q it is 'matched up' in the extensions constructed in the proof.

Lemma 3.6. *Let r and c be integer sequences of length m and n, respectively, and let M be the merge matrix of r and c. Let $i \in \operatorname{argmin}(r)$ and $j \in \operatorname{argmin}(c)$.*

(i) If $i = 1$ and $j = n$, then $p_{\lrcorner}$ dominates all paths in M, i.e. $p_{\lrcorner} \prec \mathcal{P}(M)$.

(ii) If $i = m$ and $j = 1$, then $p_{\ulcorner}$ dominates all paths in M, i.e. $p_{\ulcorner} \prec \mathcal{P}(M)$.

Proof. (i) For an illustration of this proof see Figure 4. Let q be any path in M and let $t^* := \operatorname{argmax}^\star(q)$. Let furthermore $q(t^*) = (t_r^*, t_c^*)$. We divide $p_{\lrcorner}$ and q in three consecutive parts each to show that $p_{\lrcorner}$ dominates q.

- We let $p_{\lrcorner}^1 := p_{\lrcorner}(1), \ldots, p_{\lrcorner}(t_c^* - 1)$ and $q_1 := q(1), \ldots, q(t^* - 1)$.
- We let $p_{\lrcorner}^2 := p_{\lrcorner}(t_c^*), \ldots, p_{\lrcorner}(n + t_r^* - 1)$ and $q_2 := q(t^*)$.
- We let $p_{\lrcorner}^3 := p_{\lrcorner}(n + t_r^*), \ldots, p_{\lrcorner}(m + n - 1)$ and $q_3 := q(t^* + 1), \ldots, q(l(q))$.

Since $r(1)$ is a minimum row in M, we have that for all $(k, \ell) \in [m] \times [n]$, $M[1, \ell] \le M[k, \ell]$. This implies that there is an extension e_1 of $p_{\lrcorner}^1$ of length $t^* - 1$ such that $M[e_1] \le M[q_1]$. Similarly, there is an extension e_3 of $p_{\lrcorner}^3$ of length $l(q) - t^*$ such that $M[e_3] \le M[q_3]$. Finally, let f_2 be an extension of q_2 that repeats its only element, $q(t^*)$, $n - t_c^* + t_r^*$ times. Since $M[q(t^*)]$ is the maximum element on the sequence $M[q]$ and $r(1)$ is a minimum row and $c(n)$ a minimum column in M, we have that $M[p_{\lrcorner}^2] \le M[f_2]$.

We define an extension e of $p_{\lrcorner}$ as $e := e_1 \circ p_{\lrcorner}^2 \circ e_3$ and an extension f of q as $f := q_1 \circ f_2 \circ q_3$. Note that $l(e) = l(f) = l(q) + n + t_r^* - (t_c^* + 1)$, and by the above discussion, we have that $M[e] \le M[f]$. (ii) follows from a symmetric argument. $\square$

3.2 The Split Lemma

In this section we prove the first main step towards the Merge Dominator Lemma. It is fairly intuitive that a dominating merge has to contain the minimum element of a merge matrix. (Otherwise, there is a path that cannot be dominated by that merge.) The Split Lemma states that in fact, we can split the matrix M into two smaller submatrices, one that has the minimum

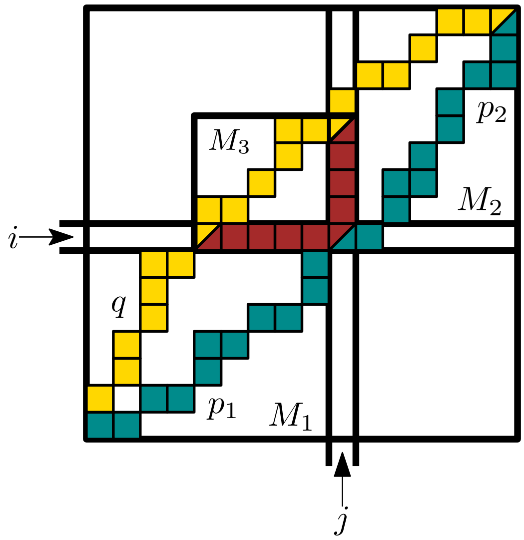


Figure 5: Situation in the proof of Lemma 3.7.

element in the top right corner, and one the has the minimum element in the bottom left corner, compute a dominating path for each of them, and paste them together to obtain a dominating path for M.

Lemma 3.7 (Split Lemma). *Let r and c be integer sequences of length m and n, respectively, and let M be the merge matrix of r and c. Let $i \in \text{argmin}(r)$ and $j \in \text{argmin}(c)$. Let $M_1 := M[1..i, 1..j]$ and $M_2 := M[i..m, j..n]$ and for all $h \in [2]$, let $p_h \in \mathcal{P}(M_t)$ be a dominating path in M_h, i.e. $p_h \prec \mathcal{P}(M_h)$. Then, $p_1 \circ p_2$ is a dominating path in M, i.e. $p_1 \circ p_2 \prec \mathcal{P}(M)$.*

Proof. Let q be any path in M. If q contains (i, j), then q has two consecutive parts, say q_1 and q_2, such that $q_1 \in \mathcal{P}(M_1)$ and $q_2 \in \mathcal{P}(M_2)$. Hence, $p_1 \prec q_1$ and $p_2 \prec q_2$, so by Lemma 2.10(v), $p_1 \circ p_2 \prec q_1 \circ q_2$.

Now let $p := p_1 \circ p_2$ and suppose q does not contain (i, j). Then, q either contains some (i, j') with $j' < j$, or some (i', j), for some $i' < i$. We show how to construct extensions of p and q that witness that p dominates q in the first case, and remark that the second case can be shown symmetrically. We illustrate this situation in Figure 5.

Suppose that q contains (i, j') with $j' < j$. We show that $p \prec q$. First, q also contains some (i', j), where $i' > i$. Let h_1 be the index of (i, j') in q, i.e. $q(h_1) = (i, j')$, and h_2 denote the index of (i', j) in q, i.e. $q(h_2) = (i', j)$. We derive the following sequences from q.

- We let $q_1 := q(1), \ldots, q(h_1)$ and $q_1^+ := q_1 \circ (i, j' + 1), \ldots, (i, j)$.
- We let $q_{12} := q(h_1), \ldots, q(h_2)$.
- We let $q_2 := q(h_2), \ldots, q(l(q))$ and $q_2^+ := (i, j), (i + 1, j), \ldots, (i', j) \circ q_2$.

Since $q_1^+ \in \mathcal{P}(M_1)$ and $p_1 \prec \mathcal{P}(M_1)$, we have that $p_1 \prec q_1^+$, similarly that $p_2 \prec q_2^+$ and considering $M_3 := M[i'..i, j..j']$, we have by Lemma 3.6(i) that $p_{12} := p_\lrcorner(M_3) = (i, j'), (i, j' + 1), \ldots, (i, j), (i + 1, j), \ldots, (i', j)$ dominates q_{12}. Consequently, we consider the following extensions of these sequences.

(I) We let $e_1 \in E(p_1)$ and $f_1 \in E(q_1^+)$ such that $l(e_1) = l(f_1)$ and $M[e_1] \leq M[f_1]$.

(II) We let $e_{12} \in E(p_{12})$, and $f_{12} \in E(q_{12})$ such that $l(e_{12}) = l(f_{12})$ and $M[e_{12}] \leq M[f_{12}]$.

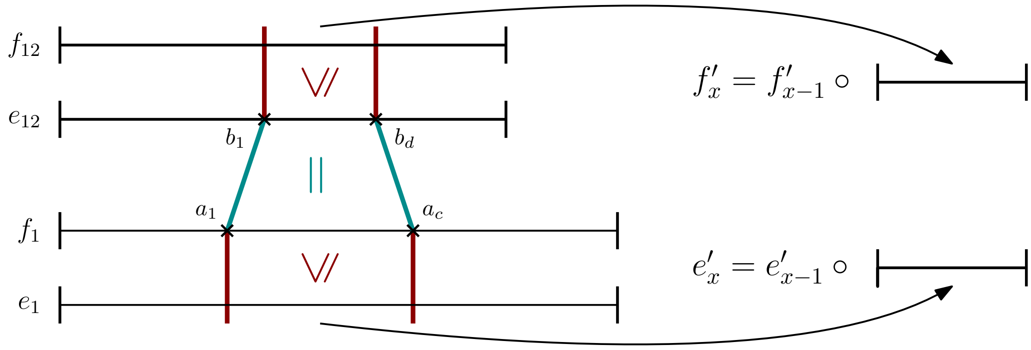


Figure 6: Constructing extensions in the proof of Lemma 3.7.

(III) We let $e_2 \in E(p_2)$, and $f_2 \in E(q_2^+)$ such that $l(e_2) = l(f_2)$ and $M[e_2] \leq M[f_2]$.

We construct extensions $e' \in E(p)$ and $f' \in E(q)$ as follows. Let z be the last index in q of any element that is matched up with (i, j) in the extensions of (II). (Following the proof of Lemma 3.6, this would mean z is the index of $\max(q_{12})$ in q.) We first construct a pair of extensions $e'_j \in E(p_1)$, and $f'_j \in E(q[1..z])$ with $l(e'_j) = l(f'_j)$ and $M[e'_j] \leq M[f'_j]$. With a symmetric procedure, we can obtain extensions of p_2 and of $q[(z+1)..l(q)]$, and use them to obtain extensions of $p = p_1 \circ p_2$ and $q = q[1..z] \circ q[(z+1)..l(q)]$ witnessing that $p \prec q$.

We give the details of the first part of the construction. Let a be the index of the last repetition in f_1 of $q(h_1 - 1)$, i.e. the index that appears just before $q(h_1) = (i, j')$ in f_1. We let $e'_{j'-1}[1..a] := e_1[1..a]$ and $f'_{j'-1}[1..a] := f_1[1..a]$. By (I), $M[e'_{j'-1}] \leq M[f'_{j'-1}]$.

For $x = j', j'+1, \ldots, j$, we inductively construct e'_x and f'_x using e'_{x-1} and f'_{x-1}, for an illustration see Figure 6. We maintain as an invariant that $l(e'_{x-1}) = l(f'_{x-1})$ and that $M[e'_{x-1}] \leq M[f'_{x-1}]$. Let $a_1, \ldots, a_c$ denote the indices of the occurrences of (i, x) in f_1, and $b_1, \ldots, b_d$ denote the indices of the occurrences of (i, x) in e_{12}. We let:

$$e'_x := e'_{x-1} \circ e_1[a_1, \ldots, a_c] \text{ and } f'_x := f'_{x-1} \circ f_{12}[b_1, \ldots, b_d], \qquad \text{if } c = d$$

$$e'_x := e'_{x-1} \circ e_1[a_1, \ldots, a_c] \circ \overbrace{e_1(a_c), \ldots, e_1(a_c)}^{d-c \text{ times}} \text{ and } f'_x := f'_{x-1} \circ f_{12}[b_1, \ldots, b_d], \qquad \text{if } c < d$$

$$e'_x := e'_{x-1} \circ e_1[a_1, \ldots, a_c] \text{ and } f'_x := f'_{x-1} \circ f_{12}[b_1, \ldots, b_d] \circ \overbrace{f_{12}(b_d), \ldots, f_{12}(b_d)}^{c-d \text{ times}}, \qquad \text{if } c > d$$

In each case, we extended e'_{x-1} and f'_{x-1} by the same number of elements; furthermore we know by (I) that for $y \in \{a_1, \ldots, a_c\}$, $M[e_1(y)] \leq M[f_1(y)]$, by choice we have that for all $y' \in \{b_1, \ldots, b_d\}$, $f_1(y) = e_{12}(y')$ and we know that $M[e_{12}(y')] \leq M[f_{12}(y')]$ by (II). Hence, $M[e'_x] \leq M[f'_x]$ in either of the above cases. In the end of this process, we have $e'_j \in E(p_1)$ and $f'_j \in E(q[1..z])$, and by construction, $l(e'_j) = l(f'_j)$ and $M[e'_j] \leq M[f'_j]$. $\square$

3.3 The Chop Lemmas

Assume the notation of the Split Lemma. If we were to apply it recursively, it only yields a size reduction whenever $(i, j) \notin \{(1, 1), (m, n)\}$. Motivated by this issue, we prove two more lemmas to deal with the cases when $(i, j) \in \{(1, 1), (m, n)\}$, and we coin them the 'Chop Lemmas'. It will turn out that when applied to typical sequences, a repeated application of these lemmas yields a dominating path in M. This insight crucially helps in arguing that the dominating path in a merge matrix can be found in *linear* time. Before we present their statements and proofs, we need another auxiliary lemma.

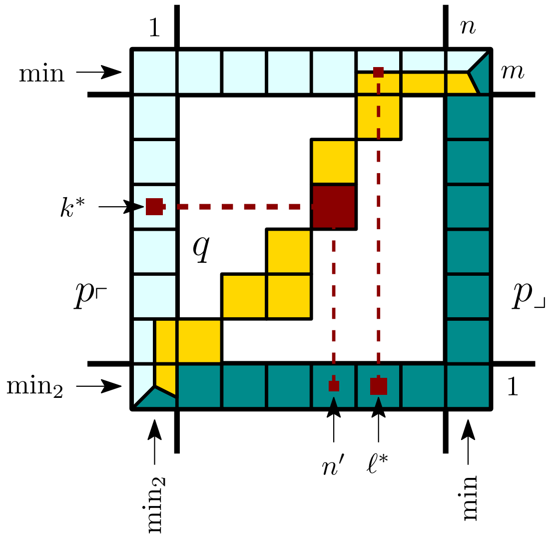


Figure 7: Situation in the first stage of the proof of Lemma 3.8(i). The row and column labeled 'min' contains the minimum element from the respective sequence, and the row and column labeled '$\min_2$' contains the minimum element among all elements except the one in the min-row or -column.

Lemma 3.8. *Let r and c be integer sequences of length m and n, respectively, and let M be the merge matrix of r and c. Let $i \in \operatorname{argmin}(r)$ and $j \in \operatorname{argmin}(c)$. Let furthermore $k \in \operatorname{argmin}(r[\{1,\ldots,m\} \setminus \{i\}])$ and $\ell \in \operatorname{argmin}(c[\{1,\ldots,n\} \setminus \{i\})$ Let $\{p^*, q^*\} = \{p_\lrcorner, p_\ulcorner\}$ such that $\max(M[p^*]) \leq \max(M[q^*])$.*

(i) If $i = m$, $j = n$, $k = 1$, and $\ell = 1$, then $p^* \prec \mathcal{P}(M)$.

(ii) If $i = 1$, $j = 1$, $k = m$, and $\ell = n$, then $p^* \prec \mathcal{P}(M)$.

Proof. (i). First, we may assume that $r(1) > r(m)$ and that $c(1) > c(n)$, otherwise we could have applied one of the cases of Lemma 3.6. We prove the lemma in two steps:

1. We show that for each path q in M, $p_\lrcorner$ or $p_\ulcorner$ (or both) dominate(s) q.
2. We show that $p_\lrcorner$ dominates $p_\ulcorner$, or vice versa, or both (depending on which case we are in).

The following claim will be useful in both steps and can be seen as a slight generalization of Lemma 3.6.

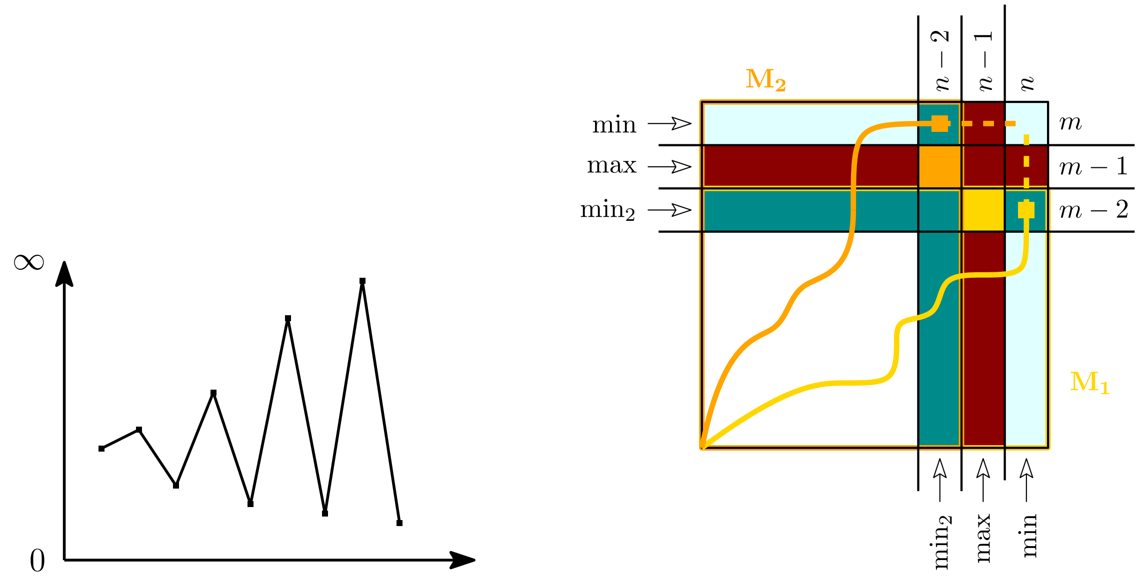
Claim 3.8.1. Let $q \in \mathcal{P}(M)$ and let $p \in \{p_\lrcorner, p_\ulcorner\}$. If $\max(M[p]) \leq \max(M[q])$, then $p \prec q$.

Proof. Suppose that $p = p_\lrcorner$, the other case is symmetric. The claim can be shown using the same argument as in Lemma 3.6, paying slight attention to the situation in which the maximum value of q is in row m, which implies that the maximum of $p_\lrcorner$ is in the same column. $\lrcorner$

We prove Step 1. For the following argument, see Figure 7. If $\max(M[q]) \geq \max(M[p_\lrcorner])$, then we conclude by Claim 3.8.1 that $p_\lrcorner \prec q$ and we are done with Step 1 of the proof. Suppose

$$\max(M[q]) < \max(M[p_\lrcorner]) \tag{3}$$

and let $\ell^* \in \operatorname{argmax}(M[p_\lrcorner])$. We may assume that $\ell^* < n$: otherwise, (3) cannot be satisfied since $n \in \operatorname{argmin}(c)$. We furthermore have that q contains (m, ℓ^*), since $p_\lrcorner$ contains $(1, \ell^*)$, and



(a) Typical sequence ending in the minimum.

(b) The basic setup in Lemma 3.10.

Figure 8: Visual aides to the proof of Lemma 3.10.

m is the only position in which r is (potentially) smaller than $r(1)$. Therefore, this is the only way in which (3) can be satisfied.

Now let $k^* \in \operatorname{argmax}(M[p_{\ulcorner}])$. As above, we may assume that $k^* < m$. Now, since q contains (m, ℓ^*), we have that q also contains (k^*, n') for some $n' < n$. It follows that

$$\max(M[p_{\ulcorner}]) = M[k^*, 1] \leq M[k^*, n'] \leq \max(M[q])$$

where the first inequality follows from the fact that $r(1) \leq r(n')$ for all $n' < n$. By Claim 3.8.1, $p_{\ulcorner}$ dominates q and we finished Step 1 of the proof. Step 2 follows from another application of Claim 3.8.1 and the lemma follows from transitivity of the domination relation. This proves (i), and (ii) follows from a symmetric argument. □

Remark 3.9. We would like to stress that up to this point, all results in this section were shown in terms of arbitrary integer sequences. For the next lemma, we require the sequences considered to be *typical sequences*. In Subsection 3.5 we will generalize the results that rely on the following lemmas to arbitrary integer sequences.

We are now ready to prove the Chop Lemmas. They come in two versions, one that is suited for the case of the bottom left submatrix after an application of the Split Lemma to M, and one for the top right submatrix. In the former case, we have that the last row is a minimum row and that the last column is a minimum column. We will prove this lemma in more detail and observe that the other case follows by symmetry with the arguments given in the following proof. For an illustration of the setting in the following lemma, see Figure 8b.

Lemma 3.10 (Chop Lemma - Bottom). *Let r and c be typical sequences of length $m \geq 3$ and $n \geq 3$, respectively, and let M be the merge matrix of r and c. Suppose that $m \in \operatorname{argmin}(r)$ and $n \in \operatorname{argmin}(c)$ and let $M_1 := M[1..(m-2), 1..n]$ and $M_2 := M[1..m, 1..(n-2)]$ and for all $h \in [2]$, let $p_h \prec \mathcal{P}(M_h)$. Let $p_1^+ := p_1 \circ (m-1, n), (m, n)$ and $p_2^+ := p_2 \circ (m, n-1), (m, n)$.*

(i) If $M[m-2, n-1] \leq M[m-1, n-2]$, then $p_1^+ \prec \mathcal{P}(M)$.

(ii) If $M[m-1, n-2] \leq M[m-2, n-1]$, then $p_2^+ \prec \mathcal{P}(M)$.

Proof. Let $s \in \{r, c\}$. Since s is a typical sequence and $l(s) \in \operatorname{argmin}(s)$, we know by Corollary 2.8 that for all $k \in [\lfloor l(s)/2 \rfloor]$,

$$l(s) - 2k + 1 \in \operatorname{argmax}(s[1..(l(s) - 2k + 1)]) \text{ and } l(s) - 2k \in \operatorname{argmin}(s[1..(l(s) - 2k)]).$$

Informally speaking, this means that the last element of s is the minimum, the $(l(s)-1)$-th element of s is the maximum, the $(l(s)-2)$-th element is 'second-smallest' element, and so on. We will therefore refer to the element at position $l(s) - 2k$ ($2k \le l(s)$) as '$\min_{k+1}(s)$' (note that the minimum is achieved when $k = 0$, hence the '+1'), and elements at position $l(s) - 2k + 1$ ($2k + 1 \le l(s) - 1$) as '$\max_k(s)$'. For an illustration of the shape of s see Figure 8a and for an illustration of the basic setting of this proof see Figure 8b. We prove (i) and remark that the argument for (ii) is symmetric.

First, we show that each path in M is dominated by at least one of p_1^+ and p_2^+.

Claim 3.10.1. Let $q \in \mathcal{P}(M)$. Then, for some $r \in [2]$, $p_r^+ \prec q$.

Proof. We may assume that q does not contain $(m-1, n-1)$: if so, we could easily obtain a path q' from q by some local replacements such that q' dominates q, since $M[m-1, n-1]$ is the maximum element of the matrix M. We may assume that q either contains $(m-1, n)$ or $(m, n-1)$. Assume that the former holds, and note that an argument for the latter case can be given analogously. Since q contains $(m-1, n)$, and since q does not contain $(m-1, n-1)$, we may assume that q contains $(m-2, n)$: if not, we can simply add $(m-2, n)$ before $(m-1, n)$ to obtain a path that dominates q (recall that n is the column indexed by the minimum of c). Now, let $q|_{M_1}$ be the restriction of q to M_1, we then have that $q = q|_{M_1} \circ (m-1, n), (m, n)$. Since p_1 dominates all paths in M_1, it dominates $q|_{M_1}$ and so $p_1^+ \prec q$. ⌟

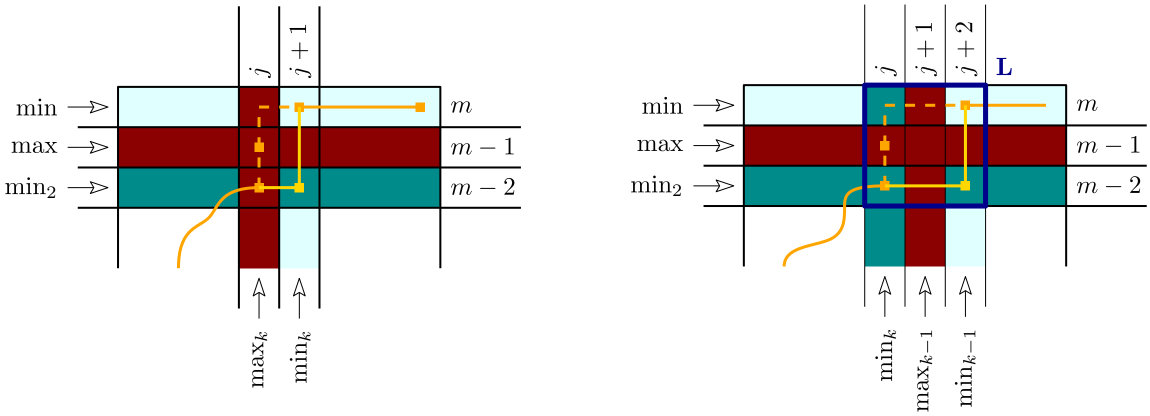
The remainder of the proof is devoted to showing that p_1^+ dominates p_2^+ which yields the lemma by Claim 3.10.1 and transitivity. To achieve that, we will show in a series of claims that we may assume that p_2 contains $(m-2, n-2)$. In particular, we show that if p_2 does not contain $(m-2, n-2)$, then there is another path in M_2 that does contain $(m-2, n-2)$ and dominates p_2.

Claim 3.10.2. We may assume that there is a unique $j \in [n-2]$ such that p_2 contains $(m-1, j)$.

Proof. Clearly, p_2 has to pass through the row $m-1$ at some point. We show that we may assume that there is a unique such point. Suppose not and let $j_1, \ldots, j_t$ be such that p_2 contains all $(m-1, j_i)$, where $i \in [t]$. By the definition of a path in a matrix, we have that $j_{i+1} = j_i + 1$ for all $i \in [t-1]$. Let p_2' be the path obtained from p_2 by replacing, for each $i \in [t-1]$, the element $(m-1, j_i)$ with the element $(m-2, j_i)$. Since $r(m-2) \le r(m-1)$ (recall that $m-1 \in \operatorname{argmax}(r)$), it is not difficult to see that p_2' dominates p_2, and clearly, p_2' satisfies the condition of the claim. ⌟

Claim 3.10.3. Let $j \in [n-3]$ be such that p_2 contains $(m-1, j)$. If $j = n - 2k + 1$ for some $k \in \mathbb{N}$ with $2k+1 \le n-1$, then there is a path p_2' that dominates p_2 and contains $(m-1, j+1)$.

Proof. For an illustration see Figure 9a. First, by Claim 3.10.2, we may assume that j is unique. Moreover, since $j = n - 2k + 1$ and $j + 1 = n - 2k + 2 = n - 2(k-1)$, we have that $c(j) = \max_k(c)$ and $c(j+1) = \min_k(c)$, respectively, and therefore $c(j+1) \le c(j)$. Hence, we may assume that the element after $(m-1, j)$ in p_2 is $(m, j+1)$: if p_2 contained (m, j) we could simply remove (m, j) from p_2 without changing the fact that p_2 is a dominating path since $M[m, j] > M[m, j+1]$. We modify p_2 as follows. We remove $(m-1, j)$, and add



(a) Situation of Claim 3.10.3.

(b) Situation of Claim 3.10.4.

Figure 9: Visualization of the arguments that lead to the conclusion that we may assume that p_2 contains $(m-2, n-2)$ in the proof of Lemma 3.10.

$(m-2, j)$ (if not already present), followed by $(m-2, j+1)$ and then $(m-1, j+1)$. For each $x \in \{M[m-2, j], M[m-2, j+1], M[m-1, j+1]\}$, we have that $x < M[m-1, j]$ (recall that $r(m-2) < r(m-1)$ and $c(j+1) < c(j)$). Hence, the resulting path dominates p_2 and it contains $(m-1, j+1)$. $\lrcorner$

Claim 3.10.4. Let $j \in [n-4]$ be such that p_2 contains $(m-1, j)$. If $j = n-2(k-1)$ for some $k \in \left[3..\left\lfloor \frac{n}{2} \right\rfloor\right]$, then there is a path p_2' that dominates p_2 and contains $(m-1, j+2)$.

Proof. For an illustration see Figure 9b. Again, by Claim 3.10.2, we may assume that j is unique. Since $j = n-2(k-1)$, we have that $c(j) = \min_k(c)$. First, if not already present, we insert $(m-2, j)$ just before $(m-1, j)$ in p_2. This does not change the fact that p_2 is a dominating path, since $M[m-2, j] < M[m-1, j]$ (recall that $r(m-2) < r(m-1)$). Next, consider the 3×3 submatrix $L := M[(m-2)..m, j..(j+2)]$. Note that L is the submatrix of M restricted to the rows $\min(r)$, $\max(r)$, and $\min_2(r)$, and the columns $\min_k(c)$, $\max_{k-1}(c)$, and $\min_{k-1}(c)$. Furthermore, we have that p_2 restricted to L is equal to $p_\ulcorner(L)$. We show that $p_\lrcorner(L)$ dominates $p_\ulcorner(L)$, from which we can conclude that we can obtain a path p_2' from p_2 that contains $(m-1, j+2)$ and dominates p_2 by replacing $p_\ulcorner(L)$ with $p_\lrcorner(L)$. By Lemma 3.8, it suffices to show that $M[m-2, j+1] \leq M[m-1, j]$, in other words, that $\max_{k-1}(c) + \min_2(r) \leq \max(r) + \min_k(c)$.

By the assumption of the lemma, we have that $M[m-2, n-1] \leq M[m-1, n-2]$, hence,

$$\max(c) + \min_2(r) \leq \max(r) + \min_2(c), \text{ and so: } \max(c) - \min_2(c) \leq \max(r) - \min_2(r).$$

Next, we have that for all $j \in [\lfloor n/2 \rfloor]$,

$$\max(c) - \min_2(c) \geq \max_j(c) - \min_{j+1}(c).$$

Putting the two together, we have that

$$\max_{k-1}(c) - \min_k(c) \leq \max(r) - \min_2(r), \text{ and so: } \max_{k-1}(c) + \min_2(r) \leq \max(r) + \min_k(c),$$

which concludes the proof of the claim. $\lrcorner$

We are now ready to conclude the proof.

Claim 3.10.5. $p_1^+ \prec p_2^+$.

Proof. By repeated application of Claims 3.10.3 and 3.10.4, we know that there is a path p_2' in M_2 that contains $(m-1, n-2)$. Furthermore, we may assume that p_2' contains $(m-2, n-2)$ as well: we can simply add this element if it is not already present; since $M[m-2, n-2] \leq M[m-1, n-2]$, this does not change the property that $p_2' \prec p_2$. Now, let p_2'' be the subpath of p_2' ending in $(m-2, n-2)$. (Note that $p_2'' \circ (m-2, n-1), (m-2, n) \in \mathcal{P}(M_1)$.) Then,

$$\begin{aligned} p_1^+ &\prec p_2'' \circ (m-2, n-1), (m-2, n), (m-1, n), (m, n) &(4)\\ &\prec p_2' \circ (m, n-1), (m, n) &(5)\\ &\prec p_2^+, &(6)\end{aligned}$$

where (4) is due to $p_1 \prec \mathcal{P}(M_1)$ and therefore $p_1 \prec p_2'' \circ (m-2, n-1), (m-2, n)$. Next (5) follows from an application of Lemma 3.8 to the 3×3-submatrix $M[(m-2)..m, (n-2)..n]$ and (6) is guaranteed since $p_2' \prec p_2$. ⌟

This concludes the proof of (i) and (ii) can be shown symmetrically. □

As the previous lemma always assumes that $m \geq 3$ and $n \geq 3$, we observe the corresponding base case which occurs when either $m \leq 2$ or $n \leq 2$. This base case is justified by the observation that in the bottom case, the last row and column of M are minimum.

Observation 3.11 (Base Case - Bottom). Let r and c be typical sequences of length m and n, respectively, and let M be the merge matrix of r and c. Suppose that $m \in \operatorname{argmin}(r)$ and $n \in \operatorname{argmin}(c)$. If $m \leq 2$ ($n \leq 2$), then[3]

$$p^* \coloneqq (1,1), (m,1), (m,2), \ldots, (m,n) \quad (p^* \coloneqq (1,1), (1,n), (2,n), \ldots, (m,n))$$

dominates $\mathcal{P}(M)$, i.e. $p^* \prec \mathcal{P}(M)$.

By symmetry, we have the following consequence of Lemma 3.10.

Corollary 3.12 (Chop Lemma - Top). *Let r and c be typical sequences of length $m \geq 3$ and $n \geq 3$, respectively, and let M be the merge matrix of r and c. Suppose that $1 \in \operatorname{argmin}(r)$ and $1 \in \operatorname{argmin}(c)$ and let $M_1 \coloneqq M[3..m, 1..n]$ and $M_2 \coloneqq M[1..m, 3..n]$ and for all $h \in [2]$, let $p_h \prec \mathcal{P}(M_h)$. Let $p_1^+ \coloneqq (1,1), (2,1) \circ p_1$ and $p_2^+ \coloneqq (1,1), (1,2) \circ p_2$.*

(i) If $M[3,2] \leq M[2,3]$, then $p_1^+ \prec \mathcal{P}(M)$.

(ii) If $M[2,3] \leq M[3,2]$, then $p_2^+ \prec \mathcal{P}(M)$.

Again, we observe the corresponding base case.

Observation 3.13 (Base Case - Top). Let r and c be typical sequences of length m and n, respectively, and let M be the merge matrix of r and c. Suppose that $1 \in \operatorname{argmin}(r)$ and $1 \in \operatorname{argmin}(c)$. If $m \leq 2$ ($n \leq 2$), then

$$p^* \coloneqq (1,1), (1,2), \ldots, (1,n), (m,n) \quad (p^* \coloneqq (1,1), (2,1), \ldots, (m,1), (m,n))$$

dominates $\mathcal{P}(M)$, i.e. $p^* \prec \mathcal{P}(M)$.

Input : Typical sequences $r(1), \ldots, r(m)$ and $c(1), \ldots, c(n)$
Output: A dominating merge of r and c

Let $i \in \operatorname{argmin}(r)$ and $j \in \operatorname{argmin}(c)$;
return `Chop-bottom` $(r[1..i],\ c[1..j]) \circ$ `Chop-top` $(r[i..m], c[j..n])$;
Procedure `Chop-bottom(`r *and* c *as above*`)`
    if $m \leq 2$ **then return** $r(1) + c(1),\ r(m) + c(1),\ r(m) + c(2),\ \ldots,\ r(m) + c(n)$;
    if $n \leq 2$ **then return** $r(1) + c(1),\ r(1) + c(n),\ r(2) + c(n),\ \ldots,\ r(m) + c(n)$;
    if $r(m-2) + c(n-1) \leq r(m-1) + c(n-2)$ **then return**
        `Chop-bottom(`$r[1..(m-2)], c$`)` $\circ\ (r(m-1) + c(n)), r(m) + c(n)$;
    if $r(m-1) + c(n-2) \leq r(m-2) + c(n-1)$ **then return**
        `Chop-bottom(`$r, c[1..(n-2)]$`)` $\circ\ (r(m) + c(n-1)), r(m) + c(n)$;
Procedure `Chop-top(`r *and* c *as above*`)`
    if $m \leq 2$ **then return** $r(1) + c(1),\ r(1) + c(2),\ \ldots,\ r(1) + c(n),\ r(m) + c(n)$;
    if $n \leq 2$ **then return** $r(1) + c(1),\ r(2) + c(1),\ \ldots,\ r(m) + c(1),\ r(m) + c(n)$;
    if $r(3) + c(2) \leq r(2) + c(3)$ **then return**
        $r(1) + c(1), (r(2) + c(1)) \circ$ `Chop-top(`$r[3..m], c$`)`;
    if $r(2) + c(3) \leq r(3) + c(2)$ **then return**
        $r(1) + c(1), (r(1) + c(2)) \circ$ `Chop-top(`$r, c[3..n]$`)`;

Algorithm 2: The Split-and-Chop Algorithm

3.4 The Split-and-Chop Algorithm

Equipped with the Split Lemma and the Chop Lemmas, we are now ready to give the algorithm that computes a dominating merge of two typical sequences. Consequently, we call this algorithm the 'Split-and-Chop Algorithm'.

Lemma 3.14. *Let r and c be typical sequences of length m and n, respectively. Then, there is an algorithm that finds in $\mathcal{O}(m+n)$ time a dominating path in the merge matrix of r and c.*

Proof. The algorithm practically derives itself from the Split Lemma (Lemma 3.7) and the Chop Lemmas (Lemma 3.10 and Corollary 3.12). However, to make the algorithm run in the claimed time bound, we are not able to construct the merge matrix of r and c. This turns out to be not necessary, as we can simply read off the crucial values upon which the recursion of the algorithm depends from the sequences directly. The details are given in Algorithm 2.

The runtime of the Chop-subroutines can be computed as $T(m+n) \leq T(m+n-2) + \mathcal{O}(1)$, which resolves to $\mathcal{O}(m+n)$. Correctness follows from Lemmas 3.7 and 3.10 and Corollary 3.12 with the base cases given in Observations 3.11 and 3.13. $\square$

3.5 Generalization to Arbitrary Integer Sequences

In this section we show how to generalize Lemma 3.14 to arbitrary integer sequences. In particular, we will show how to construct from a merge of two typical sequences $\tau(r)$ and $\tau(s)$ that dominates all of their merges, a merge of r and s that dominates all merges of r and s. The claimed result then follows from an application of Lemma 3.14. We illustrate the following construction in Figure 10.

[3]Note that in the following equation, if $m = 1$, then strictly speaking we would have that p^* repeats the element $(1, 1)$ twice which is of course not our intention. For the sake of a clear presentation though, we will ignore this slight abuse of notation, also in similar instances throughout this section.

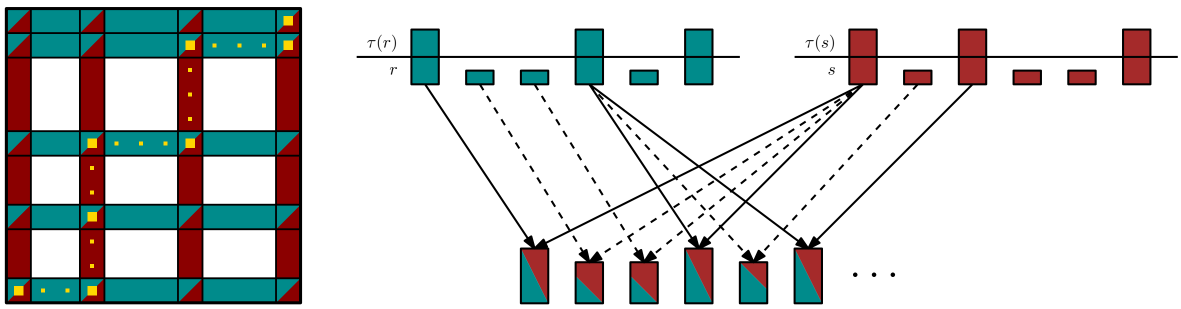


Figure 10: Illustration of the typical lift. On the left side, the view of the merge matrix M, with the rows and columns corresponding to elements of the typical sequences highlighted. Inside there, M_τ can be seen as a highlighted submatrix. The merge t' is depicted as the large yellow squares within M_τ and the small yellow squares outside of M_τ show its completion to the typical lift of t. On the right side, an illustration that does not rely on the 'matrix view'.

The Typical Lift. Let r and s be integer sequences and let $t \in \tau(r) \oplus \tau(s)$. Then, the *typical lift of* t, denoted by $\rho(t)$, is an integer sequence $\rho(t) \in r \oplus s$, obtained from t as follows. For convenience, we will consider $\rho(t)$ as a path in the merge matrix M of r and s.

Step 1. We construct $t' \in \tau(r) \boxplus \tau(s)$ such that $t' \prec t$ using Lemma 3.5. Throughout the following, consider t' to be a path in the merge matrix M_τ of $\tau(r)$ and $\tau(s)$.

Step 2. First, we initialize $\rho_t^1 := t'(1) = (1, 1)$. For $i = \{2, \ldots, l(t')\}$, we proceed inductively as follows. Let $(i_r, i_s) = t(i)$ and let $(i'_r, i'_s) = t(i-1)$. (Note that $t(i-1)$ and $t(i)$ are indices in M_τ.) Let furthermore (j_r, j_s) be the index in M corresponding to (i_r, i_s), and let (j'_r, j'_s) be the index in M corresponding to (i'_r, i'_s). Assume by induction that $\rho_t^{i-1} \in \mathcal{P}(M[1..j'_r, 1..j'_s])$. We show how to extend ρ_t^{i-1} to a path in ρ_t^i in $M[1..j_r, 1..j_s]$. Since t' is non-diagonal, we have that $(i'_r, i'_s) \in \{(i_r - 1, i_s), (i_r, i_s - 1)\}$, so one of the two following cases applies.

Case S2.1 ($i'_r = i_r - 1$ and $i'_s = i_s$). In this case, we let $\rho_t^i := \rho_t^{i-1} \circ (j'_r + 1, j_s), \ldots, (j_r, j_s)$.

Case S2.2 ($i'_r = i_r$ and $i'_s = i_s - 1$). In this case, we let $\rho_t^i := \rho_t^{i-1} \circ (j_r, j'_s + 1), \ldots, (j_r, j_s)$.

Step 3. We return $\rho(t) := \rho_t^{l(t')}$.

Furthermore, it is readily seen that the typical lift contains no diagonal steps: we obtain it from a non-diagonal path in the merge matrix of $\tau(r)$ and $\tau(s)$ by inserting vertical and horizontal paths from the merge matrix of r and s between consecutive elements. Moreover, it is computable in linear time, with Step 1 taking linear time by Lemma 3.5. We summarize in the following observation.

Observation 3.15. Let r and s be integer sequences of length m and n, respectively, and let $t \in \tau(r) \oplus \tau(s)$. Then, $\rho(t) \in r \boxplus s$, and $\rho(t)$ can be computed in time $\mathcal{O}(m+n)$.

We now show that if $t \in \tau(r) \oplus \tau(s)$ dominates all merges of $\tau(r)$ and $\tau(s)$, then the typical lift of t dominates all merges of r and s.

Lemma 3.16. *Let r and s be integer sequences and let $q \in r \oplus s$. Let $t \in \tau(r) \oplus \tau(s)$ such that $t \prec \tau(r) \oplus \tau(s)$. Then, $\rho(t) \prec q$.*

Proof. Let $t' \in \tau(r) \boxplus \tau(s)$ be the non-diagonal merge such that $t' \prec t$ used in the construction of $\rho(t)$. We argue that $\rho(t) \prec t'$. To see this, let M be the merge matrix of r and s and consider any

(j'_r, j'_s) and (j_r, j_s) as in Step 2, and suppose that $j'_s = j_s$. (Note that either $j'_s = j_s$ or $j'_r = j_r$.) As the only elements of the typical sequence of r in $[j'_r..j_r]$ are $r(j'_r)$ and $r(j_r)$, we know that either for all $h_r \in [j'_r..j_r]$, $r(j'_r) \le r(h_r) \le r(j_r)$, or for all $h_r \in [j'_r..j_r]$, $r(j'_r) \ge r(h_r) \ge r(j_r)$. Therefore, in an extension of t', we can repeat the index that yields $\max\{M[j'_r, j_s], M[j_r, j_s]\}$ sufficiently many (i.e. $j_r - j'_r$) times to ensure that the value of the extension of t' is an upper bound for all values of $\rho(t)$ in these positions.

To finish the proof, we have by Lemma 2.10(iii) that there exists a $q' \in \tau(r) \oplus \tau(s)$ such that $q' \prec q$. Since $t \prec \tau(r) \oplus \tau(s)$, we can conclude:

$$\rho(t) \prec t' \prec t \prec q' \prec q. \qquad \square$$

We wrap up and prove the Merge Dominator Lemma (Lemma 3.1), stated here in the slightly stronger form that the dominating merge is non-diagonal (which is necessary for the applications in Section 4).

Lemma 3.17 (Merge Dominator Lemma). *Let r and c be integer sequence of length m and n, respectively. There exists a dominating non-diagonal merge of r and c, i.e. an integer sequence $t \in r \boxplus c$ such that $t \prec r \oplus c$, and this dominating merge can be computed in time $\mathcal{O}(m + n)$.*

Proof. The algorithm proceeds in the following steps.

Step 1. Compute $\tau(r)$ and $\tau(c)$.

Step 2. Apply the Split-and-Chop Algorithm on input $(\tau(r), \tau(c))$ to obtain $t \prec \tau(r) \oplus \tau(c)$.

Step 3. Return the typical lift $\rho(t)$ of t.

Correctness of the above algorithm follows from Corollary 3.4 and Lemmas 3.14 and 3.16 which together guarantee that $\rho(t) \prec r \oplus c$, and by Observation 3.15, $\rho(t)$ is a non-diagonal merge, i.e. $\rho(t) \in r \boxplus c$. By Lemma 2.9, Step 1 can be done in time $\mathcal{O}(m + n)$, by Lemma 3.14, Step 2 takes time $\mathcal{O}(m+n)$ as well, and by Observation 3.15, the typical lift of t can also be computed in time $\mathcal{O}(m + n)$. Hence, the overall runtime of the algorithm is $\mathcal{O}(m + n)$. $\square$

4 Directed Width Measures of Series Parallel Digraphs

In this section, we give algorithmic consequences of the Merge Dominator Lemma. In Subsection 4.1, we provide an algorithm that computes the (weighted) cutwidth of (arc-weighted) series parallel digraphs on n vertices in time $\mathcal{O}(n^2)$. In Subsection 4.2 we provide a linear-time transformation that allows for computing the modified cutwidth of an SPD on n vertices in $\mathcal{O}(n^2)$ time, using the algorithm that computes the weighted cutwidth of an arc-weighted SPD.

4.1 Cutwidth

Recall that given a topological order $v_1, \ldots, v_n$ of a directed acyclic graph G, its cutwidth is the maximum over all $i \in [n-1]$ of the number of arcs that have their tail vertex in $\{v_1, \ldots, v_i\}$ and their head vertex in $\{v_{i+1}, \ldots, v_n\}$, and that the cutwidth of G is the minimum cutwidth over all its topological orders. We will now deal with the following computational problem.

Cutwidth of Series Parallel Digraphs

Input:	A series parallel digraph G.
Question:	What is the cutwidth of G?

Given a series parallel digraph G, we follow a bottom-up dynamic programming scheme along the decomposition tree T that yields G. Each node $t \in V(T)$ has a subgraph G_t of G associated with it, that is also series parallel. Naturally, we use the property that G_t is obtained either via series or parallel composition of the SPD's associated with its two children.

To make this problem amenable to be solved using merges of integer sequences, we define the following notion of a cut-size sequence of a topological order of a directed acyclic graph which records for each position in the order, how many arcs cross it.

Definition 4.1 (Cut-Size Sequence). Let G be a directed acyclic graph on n vertices and let $\pi \in \Pi(G)$ be a topological order of G. The sequence $x(1), \ldots, x(n-1)$, where for $i \in [n-1]$,

$$x(i) = |\{uv \in A(G) \mid \pi(u) \leq i \land \pi(v) > i\}|,$$

is the *cut-size sequence* of π, and denoted by $\sigma(\pi)$. For a set of topological orders $\Pi' \subseteq \Pi(G)$, we let $\sigma(\Pi') \coloneqq \{\sigma(\pi) \mid \pi \in \Pi'\}$.

Throughout the remainder of this section, we slightly abuse notation: If G_1 and G_2 are SPD's that are being composed with a series composition, and $\pi_1 \in \Pi(G_1)$ and $\pi_2 \in \Pi(G_2)$, then we consider $\pi = \pi_1 \circ \pi_2$ to be the concatenation of the two topological orders where $t_2 = s_1$ only appears *once* in π.

We first argue via two simple observations that when computing the cutwidth of a series parallel digraph G by following its decomposition tree in a bottom up manner, we only have to keep track of a set of topological orders that induce a set of cut-size sequences that dominate all cut-size sequences of G.

Observation 4.2. Let G be a DAG and $\pi, \lambda \in \Pi(G)$. If $\sigma(\pi) \prec \sigma(\lambda)$, then $\mathsf{cutw}(\pi) \leq \mathsf{cutw}(\lambda)$.

This is simply due to the fact that $\sigma(\pi) \prec \sigma(\lambda)$ implies that $\max(\sigma(\pi)) \leq \max(\sigma(\lambda))$. Next, if G is obtained from G_1 and G_2 via series or parallel composition, and we have $\pi_1, \lambda_1 \in \Pi(G_1)$ such that $\sigma(\pi_1) \prec \sigma(\lambda_1)$, then it is always beneficial to choose π_1 over λ_1, and π_1 can be disregarded.

Observation 4.3. Let G be an SPD that is obtained via series or parallel composition from SPD's G_1 and G_2. Let $\pi_1, \lambda_1 \in \Pi(G_1)$ be such that $\sigma(\pi_1) \prec \sigma(\lambda_1)$. Let $\pi, \lambda \in \Pi(G)$ be such that $\pi|_{V(G_1)} = \pi_1$, $\lambda|_{V(G_1)} = \lambda_1$, and for all $v \in V(G_2)$, $\pi(v) = \lambda(v)$. Then, $\sigma(\pi) \prec \sigma(\lambda)$.

The previous observation is justified as follows. Let $\sigma(\pi) = x(1), \ldots, x(n-1)$ and $\sigma(\lambda) = y(1), \ldots, y(n-1)$. Then, for each $i \in [n-1]$, the arcs of G_2 contribute equally to the values $x(i)$ and $y(i)$ (in particular since G_1 and G_2 are arc-disjoint). Therefore, we can use extensions of $\sigma(\pi_1)$ and $\sigma(\lambda_1)$ that witnesses that $\sigma(\pi_1) \prec \sigma(\lambda_1)$ to construct extensions of $\sigma(\pi)$ and $\sigma(\lambda)$ that witness that $\sigma(\pi) \prec \sigma(\lambda)$.

The following lemma states that the cut-size sequences of an SPD G can be computed by pairwise concatenation or non-diagonal merging (depending on whether G is obtained via series or parallel composition) of the two smaller SPD's that G is obtained from. Intuitively speaking, the reason why we can only consider *non-diagonal* merges is the following. When G is obtained from G_1 and G_2 via parallel composition, then each topological order of G can be considered the 'merge' of a topological order of G_1 and one of G_2, where each position (apart from the first and the last) contains a vertex *either* from G_1 *or* from G_2. Now, in a merge of a cut-size sequence of G_1 with a cut-size sequence of G_2, a diagonal step would essentially mean that in some position, we insert both a vertex from G_1 and a vertex of G_2; this is of course not possible.

Lemma 4.4. *Let G_1 and G_2 be SPD's. Then the following hold.*

(i) $\sigma(\Pi(G_1 \vdash G_2)) = \sigma(\Pi(G_1)) \odot \sigma(\Pi(G_2))$.

(ii) $\sigma(\Pi(G_1 \perp G_2)) = \sigma(\Pi(G_1)) \boxplus \sigma(\Pi(G_2))$.

Proof. (i). Let $\sigma(\pi) \in \sigma(\Pi(G_1 \vdash G_2))$ be such that π is a topological order of $G_1 \vdash G_2$. Then, π consists of two contiguous parts, namely $\pi_1 := \pi|_{V(G_1)} \in \Pi(G_1)$ followed by $\pi_2 := \pi|_{V(G_2)} \in \Pi(G_2)$. Since there are no arcs from $V(G_1) \setminus \{t_1\}$ to $V(G_2) \setminus \{s_2\}$, we have that $\sigma(\pi) = \sigma(\pi_1) \circ \sigma(\pi_2) \in \sigma(\Pi(G_1)) \odot \sigma(\Pi(G_2))$. The other inclusion follows similarly.

(ii). Let $\sigma(\pi) \in \sigma(\Pi(G_1 \perp G_2))$ be such that π is a topological order of $G_1 \perp G_2$. Let $\pi_1 := \pi|_{V(G_1)}$ and $\pi_2 := \pi|_{V(G_2)}$. It is clear that $\pi_1 \in \Pi(G_1)$ and that $\pi_2 \in \Pi(G_2)$. Let $\sigma(\pi) = x(1), \ldots, x(n-1)$, $\sigma(\pi_1) = y_1(1), \ldots, y_1(n_1 - 1)$, and $\sigma(\pi_2) = y_2(1), \ldots, y_2(n_2 - 1)$. For any $i \in \{1, \ldots, n-1\}$, let i_1 be the maximum index such that $\pi(\pi_1^{-1}(i_1)) \leq i$, and define i_2 accordingly. Then, the set of arcs that cross the cut between positions i and $i+1$ in π is the union of the set of arcs crossing the cut between positions i_1 and $i_1 + 1$ in π_1 and the set of arcs crossing the cut between positions i_2 and $i_2 + 1$ in π_2. Since G_1 and G_2 are arc-disjoint, this means that $x(i) = y_1(i_1) + y_2(i_2)$. Together with the observation that each vertex at position $i + 1 < n$ in π is *either* from G_1 *or* from G_2, we have that

$$x(i+1) \in \{y_1(i_1+1) + y_2(i_2), y_1(i_1) + y_2(i_2+1)\},$$

in other words, we have that $\sigma(\pi) \in \sigma(\pi_1) \boxplus \sigma(\pi_2) \subseteq \sigma(\Pi(G_1)) \boxplus \sigma(\Pi(G_2))$. The other inclusion can be shown similarly, essentially using the fact that we are only considering non-diagonal merges. □

We now prove the crucial lemma of this section which states that we can compute a dominating cut-size sequence of an SPD G from dominating cut-size sequences of the smaller SPD's that G is obtained from. For technical reasons, we assume in the following lemma that G has no parallel arcs, which does not affect the algorithm presented in this section.

Lemma 4.5. *Let G be an SPD without parallel arcs. Then there is a topological order π^* of G such that $\sigma(\pi^*)$ dominates all cut-size sequences of G. Moreover, the following hold. Let G_1 and G_2 be SPD's and for $r \in [2]$, let π_r^* be a topological order of G_r such that $\sigma(\pi_r^*)$ dominates all cut-size sequences of G_r.*

(i) If $G = G_1 \vdash G_2$, then $\pi^* = \pi_1^* \circ \pi_2^*$.

(ii) If $G = G_1 \perp G_2$, then π^* can be found as the topological order of G such that $\sigma(\pi^*)$ dominates $\sigma(\pi_1^*) \boxplus \sigma(\pi_2^*)$.

Proof. We prove the lemma by induction on the number of vertices of G. If $|V(G)| = 2$, then the claim is trivially true (there is only one topological order). Suppose that $|V(G)| =: n > 2$. Since $n > 2$ and G has no parallel arcs, we know that G can be obtained from two SPD's G_1 and G_2 via series or parallel composition with $|V(G_1)| =: n_1 < n$ and $|V(G_2)| =: n_2 < n$. By the induction hypothesis, for $r \in [2]$, there is a unique topological order π_r^* such that $\sigma(\pi_r^*)$ dominates all cut-size sequences of G_r.

Suppose $G = G_1 \vdash G_2$. Since $\sigma(\pi_1^*)$ dominates all cut-size sequences of G_1 and $\sigma(\pi_2^*)$ dominates all cut-size sequences of G_2, we can conclude using Lemma 2.10(v) that $\sigma(\pi_1^*) \circ \sigma(\pi_2^*)$ dominates $\sigma(\Pi(G_1)) \odot \sigma(\Pi(G_2))$ which together with Lemma 4.4(i) allows us to conclude that $\sigma(\pi_1^*) \circ \sigma(\pi_2^*) = \sigma(\pi_1^* \circ \pi_2^*)$ dominates all cut-size sequences of G. This proves (i).

Suppose that $G = G_1 \perp G_2$, and let π^* be a topological order of G such that $\sigma(\pi^*)$ dominates $\sigma(\pi_1^*) \boxplus \sigma(\pi_2^*)$. We show that $\sigma(\pi^*)$ dominates $\sigma(\Pi(G))$. Let $\pi \in \Pi(G)$. By Lemma 4.4(ii), there exist topological orders $\pi_1 \in \Pi(G_1)$ and $\pi_2 \in \Pi(G_2)$ such that $\sigma(\pi) \in \sigma(\pi_1) \boxplus \sigma(\pi_2)$.

In other words, there are extensions e_1 of $\sigma(\pi_1)$ and e_2 of $\sigma(\pi_2)$ of the same length such that $\sigma(\pi) = e_1 + e_2$. For $r \in [2]$, since $\sigma(\pi_r^*) \prec \sigma(\pi_r)$, we have that $\sigma(\pi_r^*) \prec e_r$. By Lemma 2.10(ii),[4] there exists some $f \in \sigma(\pi_1^*) \oplus \sigma(\pi_2^*)$ such that $f \prec e_1 + e_2$, and by Lemma 3.5, there is some $f' \in \sigma(\pi_1^*) \boxplus \sigma(\pi_2^*)$ such that $f' \prec f$. Since $\sigma(\pi^*) \prec \sigma(\pi_1^*) \boxplus \sigma(\pi_2^*)$, we have that $\sigma(\pi^*) \prec f'$, and hence (ii) follows:

$$\sigma(\pi^*) \prec f' \prec f \prec e_1 + e_2 = \sigma(\pi). \qquad \square$$

We are now ready to prove the first main result of this section.

Theorem 4.6. *Let G be an SPD on n vertices. There is an algorithm that computes in time $\mathcal{O}(n^2)$ the cutwidth of G, and outputs a topological ordering that achieves the upper bound.*

Proof. We may assume that G has no parallel arcs; if so, we simply subdivide all but one of the parallel arcs. This neither changes the cutwidth, nor the fact that G is series parallel. We can therefore apply Lemma 4.5 on G in the correctness proof later.

We use the algorithm of Valdes et al. [20] to compute in time $\mathcal{O}(n + |A(G)|)$ a decomposition tree T that yields G, see Theorem 2.13. We process T in a bottom-up fashion, and at each node $t \in V(T)$, compute a topological order π_t of G_t, the series parallel digraph associated with node t, such that $\sigma(\pi_t)$ dominates all cut-size sequences of G_t. Let $t \in V(T)$.

Case 1 (t is a leaf node). In this case, G_t is a single arc and there is precisely one topological order of G_t; we return that order.

Case 2 (t is a series node with left child ℓ and right child r). In this case, we look up π_ℓ, a topological order such that $\sigma(\pi_\ell)$ dominates all cut-size sequences of G_ℓ, and π_r, a topological order such that $\sigma(\pi_r)$ dominates all cut-size sequences of G_r. Following Lemma 4.5(i), we return $\pi_\ell \circ \pi_r$.

Case 3 (t is a parallel node with left child ℓ and right child r). In this case, we look up π_ℓ and π_r as in Case 2, and we compute π_t such that $\sigma(\pi_t)$ dominates $\sigma(\pi_\ell) \boxplus \sigma(\pi_r)$ using the Merge Dominator Lemma (Lemma 3.17). Following Lemma 4.5(ii), we return π_t.

Finally, we return $\pi_{\mathfrak{r}}$, the topological order of $G_{\mathfrak{r}} = G$, where $\mathfrak{r}$ is the root of T. Observations 4.2 and 4.3 ensure that it is sufficient to compute in each of the above cases a set $\Pi_t^* \subseteq \Pi(G_t)$ with the following property. For each $\pi_t \in \Pi(G_t)$, there is a $\pi_t^* \in \Pi_t^*$ such that $\sigma(\pi_t^*) \prec \sigma(\pi_t)$. By Lemma 4.5, we know that we can always find such a set of size one which is precisely what we compute in each of the above cases. Correctness of the algorithm follows. Since T has $\mathcal{O}(n)$ nodes and each of the above cases can be handled in at most $\mathcal{O}(n)$ time by Lemma 3.17, we have that the total runtime of the algorithm is $\mathcal{O}(n^2)$. $\square$

Our algorithm in fact works for the more general problem of computing the *weighted* cutwidth of a series parallel digraph which we now define formally.

Definition 4.7. Let G be a directed acyclic graph and $\omega \colon A(G) \to \mathbb{N}$ be a weight function. For a topological order $\pi \in \Pi(G)$ of G, the *weighted cutwidth of* (π, ω) is defined as

$$\mathsf{wcutw}(\pi, \omega) := \max_{i \in [n-1]} \sum_{\substack{vw \in A(G)\\ \pi(v) \le i, \pi(w) > i}} \omega(vw),$$

and the *weighted cutwidth of* (G, ω) is $\mathsf{wcutw}(G, \omega) := \min_{\pi \in \Pi(G)} \mathsf{wcutw}(\pi, \omega)$.

[4] Take $r = e_1$, $s = e_2$, $r_0 = \sigma(\pi_1)$, and $s_0 = \sigma(\pi_2)$.

The corresponding computational problem is defined as follows.

Weighted Cutwidth of Series Parallel Digraphs	
Input:	A series parallel digraph G and an arc-weight function $\omega\colon A(G) \to \mathbb{N}$.
Question:	What is the weighted cutwidth of (G, ω)?

Corollary 4.8. Let G be an SPD on n vertices and $\omega\colon A(G) \to \mathbb{N}$ an arc-weight function. There is an algorithm that computes in time $\mathcal{O}(n^2)$ the weighted cutwidth of (G, ω), and outputs a topological ordering that achieves the upper bound.

4.2 Modified Cutwidth

We now show how to use the algorithm for computing the weighted cutwidth of series parallel digraphs from Corollary 4.8 to give an algorithm that computes the *modified cutwidth* of a series parallel digraph on n vertices in time $\mathcal{O}(n^2)$. Recall that given a topological order $v_1, \ldots, v_n$ of a directed acyclic graph G, its modified cutwidth is the maximum over all $i \in [n-1]$ of the number of arcs that have their tail vertex in $\{v_1, \ldots, v_{i-1}\}$ and their head vertex in $\{v_{i+1}, \ldots, v_n\}$, and that the modified cutwidth of G is the minimum modified cutwidth over all its topological orders. We are dealing with the following computational problem.

Modified Cutwidth of Series Parallel Digraphs	
Input:	A series parallel digraph G.
Question:	What is the modified cutwidth of G?

To solve this problem, we will provide a transformation which allows for applying the algorithm for the Weighted Cutwidth of SPD's problem to compute the modified cutwidth. We would like to remark that this transformation is similar to one provided in [6], however some modifications are necessary to ensure that the digraph resulting from the transformation is an SPD.

Theorem 4.9. *Let G be an SPD on n vertices. There is an algorithm that computes in time $\mathcal{O}(n^2)$ the modified cutwidth of G, and outputs a topological ordering of G that achieves the upper bound.*

Proof. We give a transformation that enables us to solve Modified Cutwidth of SPD's with help of an algorithm that solves Weighted Cutwidth of SPD's.

Let $(G, (s, t))$ be an SPD on n vertices and m arcs. Again, we assume that G has no parallel arcs; if so, we simply subdivide all but one of the parallel arcs. This does not change the (modified) cutwidth, and keeps a digraph series parallel. We construct another digraph G' and an arc-weight function $\omega\colon A(G') \to \mathbb{N}$ as follows. For each vertex $v \in V(G) \setminus \{s, t\}$, we add to G' two vertices v_{in} and v_{out}. We add s and t to G' and write s as s_{out} and t as t_{in}. We add the following arcs to G'. First, for each $v \in V(G)$, we add an arc (v_{in}, v_{out}) and we let $\omega((v_{in}, v_{out})) := m + 1$. Next, for each arc $(v, w) \in A(G)$, we add an arc (v_{out}, w_{in}) to G' and we let $\omega((v_{out}, w_{in})) := 1$. For an illustration see Figure 11.

We observe that the size of G' is linear in the size of G, and then prove that if G' is obtained from applying the above transformation to a series parallel digraph, then G' is itself an SPD.

Observation 4.9.1. Let G and G' be as above. Then, $n' := |V(G')| \leq 2|V(G)|$ and $|A(G')| \leq |A(G)| + |V(G)|$.

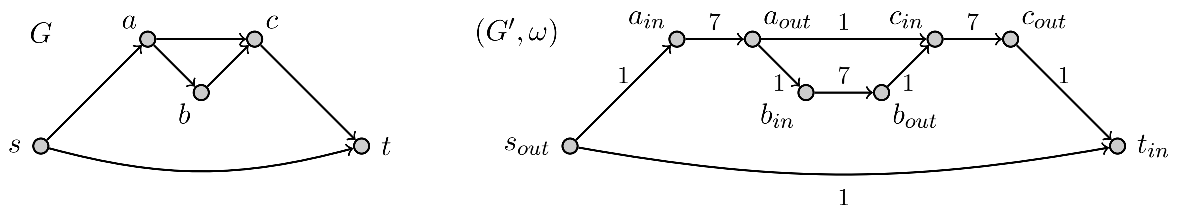


Figure 11: Illustration of the transformation given in the proof of Theorem 4.9. Note that in this case, $m = 6$, so the arcs between vertices v_{in} and v_{out} have weight 7.

Claim 4.9.2. If G is a series parallel digraph, then G' as constructed above is an SPD.

Proof. We prove the claim by induction on n, the number of vertices of G. For the base case when $n = 2$, we have that G is a single arc in which case G' is a single arc as well. Now suppose $n > 2$. Since $n > 2$, G is obtained from two series parallel digraphs G_1 and G_2 via series or parallel composition. Since G has no parallel arcs, we can use the induction hypothesis to conclude that the graphs G_1' and G_2' obtained via our construction are series parallel. Now, if $G = G_1 \perp G_2$, then it is immediate that G' is series parallel. If $G = G_1 \vdash G_2$, then we have that in G', the vertex that was constructed since t_1 and s_2 were identified, call this vertex x, got split into two vertices x_{in} and x_{out} with a directed arc of weight $m+1$ pointing from x_{in} to x_{out}. Call the series parallel digraph consisting only of this arc $(X, (x_{in}, x_{out}))$. We now have that $G' = G_1' \vdash X \vdash G_2'$, so G' is series parallel in this case as well. ⌟

We are now ready to prove the correctness of this transformation. To do so, we will assume that we are given an integer k and we want to decide whether the modified cutwidth of G is at most k.

Claim 4.9.3. If G has modified cutwidth at most k, then G' has weighted cutwidth at most $m + k + 1$.

Proof. Take a topological ordering π of G such that $\mathsf{mcutw}(\pi) \le k$. We obtain π' from π by replacing each vertex $v \in V(G) \setminus \{s, t\}$ by v_{in} followed directly by v_{out}. Clearly, this is a topological order of G'. We show that the weighted cutwidth of this ordering is at most $m + k + 1$.

Let $i \in [n' - 1]$ and consider the cut between position i and $i + 1$ in π'. We have to consider two cases. In the first case, there is some $v \in V(G)$ such that $\pi'^{-1}(i) = v_{in}$ and $\pi'^{-1}(i+1) = v_{out}$. Then, there is an arc of weight $m + 1$ from v_{in} to v_{out} crossing this cut, and some other arcs of the form (u_{out}, w_{in}) for some arc $(u, w) \in A(G)$. All these arcs cross position $\pi(v)$ in π, so since $\mathsf{mcutw}(\pi) \le k$, there are at most k of them. Furthermore, for each such arc we have that $\omega((u_{out}, w_{in})) = 1$ by construction, so the total weight of this cut is at most $m + k + 1$.

In the second case, we have that $\pi'^{-1}(i) = v_{out}$ and $\pi'^{-1}(i + 1) = w_{in}$ for some $v, w \in V(G)$, $v \neq w$. By construction, we have that $\pi(w) = \pi(v) + 1$. Hence, any arc crossing the cut between i and $i + 1$ in π' is of one of the following forms.

(i) It is (x_{out}, y_{in}) for some $(x, y) \in A(G)$ with $\pi(x) < \pi(v)$ and $\pi(y) > \pi(v)$, or

(ii) it is (x_{out}, y_{in}) for some $(x, y) \in A(G)$ with $\pi(x) < \pi(w)$ and $\pi(y) > \pi(w)$, or

(iii) it is (v_{out}, w_{in}).

Since $\mathsf{mcutw}(G) \le k$, there are at most k arcs of the first and second type, and since G has no parallel arcs, there is at most one arc of the third type. By construction, all these arcs have weight one, so the total weight of this cut is $2k + 1 \le m + k + 1$. ⌟

Claim 4.9.4. If G' has weighted cutwidth at most $m+k+1$, then G has modified cutwidth at most k.

Proof. Let π' be a topological order of G' such that $\mathsf{wcutw}(\pi',\omega) \leq m+k+1$. First, we claim that for all $v \in V(G) \setminus \{s,t\}$, we have that $\pi'(v_{out}) = \pi'(v_{in})+1$. Suppose not, for some vertex v. If we have that $\pi'(v_{in}) < \pi'(w_{in}) < \pi'(v_{out})$ for some $w \in V(G) \setminus \{s,t\}$ and $w \neq v$, then the cut between $\pi'(w_{in})$ and $\pi'(w_{in})+1$ has weight at least $2m+2$: the two arcs (v_{in}, v_{out}) and (w_{in}, w_{out}) cross this cut, and they are of weight $m+1$ each. Similarly, if $\pi'(v_{in}) < \pi'(w_{out}) < \pi'(v_{out})$, then the cut between $\pi'(w_{out})-1$ and $\pi'(w_{out})$ has weight at least $2m+2$. Since $2m+2 > m+k+1$, we have a contradiction in both cases.

We define a linear ordering π of G as follows. We let $\pi(s) := 1$, $\pi(t) := n$, and for all $v, w \in V(G) \setminus \{s,t\}$, we have $\pi(v) < \pi(w)$ if and only if $\pi'(v_{in}) < \pi'(w_{in})$. It is clear that π is a topological ordering of G; we show that π has modified cutwidth at most k. Consider an arc (x,y) that crosses a vertex v in π, i.e. we have that $\pi(x) < \pi(v) < \pi(y)$. We have just argued that $\pi'(v_{out}) = \pi'(v_{in})+1$, so we have that the arc (x_{out}, y_{in}) crosses the cut between v_{in} and v_{out} in π'. Recall that there is an arc of weight $m+1$ from v_{in} to v_{out}, so since $\mathsf{wcutw}(\pi',\omega) \leq m+k+1$, we can conclude that in π, there are at most $(m+k+1)-(m-1) = k$ arcs crossing the vertex v in π. ⌟

Now, to compute the modified cutwidth of G, we run the above described transformation to obtain (G', ω), and compute a topological order that gives the smallest weighted cutwidth of (G', ω) using Corollary 4.8. We can then follow the argument given in the proof of Claim 4.9.4 to obtain a topological order for G that gives the smalles modified cutwidth of G.

By Claim 4.9.2, G' is an SPD, so we can indeed apply the algorithm of Corollary 4.8 to solve the instance (G', ω). Correctness follows from Claims 4.9.3 and 4.9.4. By Observation 4.9.1, $|V(G')| = \mathcal{O}(|V(G)|) = \mathcal{O}(n)$, and clearly, (G', ω) can be constructed in time $\mathcal{O}(|V(G)|+|A(G)|)$; so the overall runtime of this procedure is at most $\mathcal{O}(n^2)$. ☐

5 Conclusions

In this paper, we obtained a new technical insight in a now over a quarter century old technique, namely the use of typical sequences. The insight led to new polynomial time algorithms. Since its inception, algorithms based on typical sequences give the best asymptotic bounds for linear time FPT algorithms for treewidth and pathwidth, as functions of the target parameter. It still remains a challenge to improve upon these bounds ($2^{O(pw^2)}$, respectively $2^{O(tw^3)}$), or give non-trivial lower bounds for parameterized pathwidth or treewidth. Possibly, the Merge Dominator Lemma can be helpful to get some progress here.

As other open problems, we ask whether there are other width parameters for which the Merge Dominator Lemma implies polynomial time or XP algorithms, or whether such algorithms exist for other classes of graphs. For instance, for which width measures can we give XP algorithms when parameterized by the treewidth of the input graph?

References

[1] Ernst Althaus and Sarah Ziegler. Optimal tree decompositions revisited: A simpler linear-time FPT algorithm, 2019. arXiv:1912.09144.

[2] Eyal Amir. Approximation algorithms for treewidth. *Algorithmica*, 56(4):448–479, 2010.

[3] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[4] Hans L. Bodlaender, Leizhen Cai, Jianer Chen, Michael R. Fellows, Jan Arne Telle, and Dániel Marx. Open problems in parameterized and exact computation – IWPEC 2006. Technical Report UU-CS-2006-052, Department of Information and Computing Sciences, Utrecht University, 2006.

[5] Hans L Bodlaender, Pål Grønås Drange, Markus S Dregi, Fedor V Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016.

[6] Hans L. Bodlaender, Michael R. Fellows, and Dimitrios M. Thilikos. Derivation of algorithms for cutwidth and related graph layout parameters. *Journal of Computer and System Sciences*, 75(4):231–244, 2009.

[7] Hans L. Bodlaender, Jens Gustedt, and Jan Arne Telle. Linear-time register allocation for a fixed number of registers. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 1998*, pages 574–583. ACM/SIAM, 1998.

[8] Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996.

[9] Hans L. Bodlaender and Dimitrios M. Thilikos. Constructive linear time algorithms for branchwidth. In *Proceedings 24th International Colloquium on Automata, Languages and Programming, ICALP 1997*, volume 1256 of *Lecture Notes in Computer Science (LNCS)*, pages 627–637. Springer, 1997.

[10] Hans L. Bodlaender and Dimitrios M. Thilikos. Computing small search numbers in linear time. In *Proceedings of the 1st International Workshop on Parameterized and Exact Computation, IWPEC 2004*, volume 3162 of *Lecture Notes in Computer Science (LNCS)*, pages 37–48. Springer, 2004.

[11] Mikolaj Bojanczyk and Michal Pilipczuk. Optimizing tree decompositions in MSO. In Heribert Vollmer and Brigitte Vallée, editors, *Proceedings of the 34th Symposium on Theoretical Aspects of Computer Science, STACS 2017*, volume 66 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:13, 2017.

[12] Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM Journal on Computing*, 38(2):629–657, 2008.

[13] Martin Fürer. Faster computation of path-width. In *Proceedings 27th International Workshop on Combinatorial Algorithms, IWOCA 2016*, volume 9843 of *Lecture Notes in Computer Science (LNCS)*, pages 385–396. Springer, 2016.

[14] Jens Lagergren. Efficient parallel algorithms for graphs of bounded tree-width. *Journal of Algorithms*, 20(1):20–44, 1996.

[15] Jens Lagergren and Stefan Arnborg. Finding minimal forbidden minors using a finite congruence. In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming, ICALP 1991*, volume 510 of *Lecture Notes in Computer Science (LNCS)*, pages 532–543. Springer, 1991.

[16] Bruce A. Reed. Finding approximate separators and computing tree width quickly. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, STOC 1992*, pages 221–228. ACM, 1992.

[17] Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

[18] Dimitrios M. Thilikos, Maria J. Serna, and Hans L. Bodlaender. Cutwidth I: A linear time fixed parameter algorithm. *Journal of Algorithms*, 56(1):1–24, 2005.

[19] Dimitrios M. Thilikos, Maria J. Serna, and Hans L. Bodlaender. Cutwidth II: algorithms for partial w-trees of bounded degree. *Journal of Algorithms*, 56(1):25–49, 2005.

[20] Jacobo Valdes, Robert E. Tarjan, and Eugene L. Lawler. The recognition of series-parallel digraphs. *SIAM Journal on Computing*, 11(2):298–313, 1982.