Typical Sequences Revisited — Computing Width Parameters of Graphs*

Hans L. Bodlaender[1], Lars Jaffke[2], and 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`

October 1, 2019

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 scheduling of straight-line code (a basic block) to minimize the number of registers is solvable in quadratic time for series-parallel digraphs. Previously, a polynomial-time algorithm was known only for trees [Sethi and Ullman, JACM 1970]. We also show that the Cutwidth, Modified Cutwidth, and Vertex Separation problems can be solved in $\mathcal{O}(n^2)$ time for series parallel digraphs on n vertices.

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.

*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).

The notion of typical sequences was introduced independently in 1991 by Lagergren and Arnborg [16] 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 time bounds. The latter result is a main subroutine in Bodlaender's linear time algorithm [2] for treewidth-k. If one analyses the running time of Bodlaender's algorithm for treewidth or pathwidth $\le k$, 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]. 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. [1, 5, 11, 12, 13, 15, 17, 18]), 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 [16] and [8]. Using our Merge Dominator Lemma we can go one step towards the latter, as follows.

The algorithms of Lagergren and Arnborg [16] 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 for several other problems. The first one we will discuss is an important problem in compiler optimization, specifically

[1] This approach was later used in several follow up results to obtain explicit constructive parameterized algorithms for other graph width measures, like cutwidth [21, 22], branchwidth [9], different types of search numbers like linear width [10], and directed vertex separation number [7].

a problem related to register allocation: we are given a set of expressions (a “basic block” or “straight-line code”) that have certain dependencies among each other and the task is to find a sequence of executing these expressions, respecting the dependencies, such that the number of used registers is minimized. The dependencies among these expressions form an acyclic digraph and any allowed schedule is a topological ordering. Hence, the problem is equivalent to what is known as computing the Vertex Separation Number on acyclic digraphs [7]. The problem was shown to be NP-hard by Sethi [19] while Kessler [14] gave a $2^{\mathcal{O}(n)}$ time exact algorithm, improving over the $n^{\mathcal{O}(n)}$ naive brute-force approach. Sethi and Ullman [20] showed in 1970 that the problem is linear time solvable if the acyclic digraph is a tree. Using the Merge Dominator Lemma, after almost 50 years, we can add an $\mathcal{O}(n^2)$ time algorithm for the class of *series parallel digraphs* to this.

Moreover, we also provide the first polynomial-time algorithms on series-parallel digraphs for the related Cutwidth and Modified Cutwidth problems, where again (as in [6]) we compute the solutions only among topological orders of the input acyclic digraph. Note that the vertex separation number of a digraph is equal to its pathwidth plus one, when we consider all possible orders of the input digraph.

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 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’. 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 $n-2$. Some additional technical ideas are added to obtain the algorithms for Modified Cutwidth and Vertex Separation Number for 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, directed modified cutwidth, and directed vertex separation number 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 sometimes 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, $s[X] \coloneqq s(x_1), \ldots, s(x_m)$. For $m \le n$, we call $s[\{1, \ldots, m\}]$ a *prefix* of s and $s[\{m, m+1, \ldots, n\}]$ a *suffix* of s.

Let A be a set. A *matrix* $M \in A^{m \times n}$ 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 sets $\{a, a+1, \ldots, a+x\} \subseteq [m]$ and $\{b, b+1, \ldots, b+y\} \subseteq [n]$, we use the shorthand '$M[a..(a+x), b..(b+y)]$' for $M[\{a, a+1, \ldots, a+x\}, \{b, b+1, \ldots, b+y\}]$. For a sequence $s(1), s(2), \ldots, s(\ell)$ of indices of a matrix M, we let

$$M[s] \coloneqq M[s(1)], M[s(2)], \ldots, M[s(\ell)] \qquad (1)$$

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

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

be the set of indices at whose positions there are the minimum and maximum element of s, respectively. Whenever we write $i \in \operatorname{argmin}(s)$ ($j \in \operatorname{argmax}(j)$), then the choice of i 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

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

For a subset of indices $X \subseteq [n]$, we define $\min_X(s)$ to be the minimum element of s among the ones whose index is in X, and we define $\operatorname{argmin}_X(s)$ accordingly:

$$\min_X(s) \coloneqq \min_{i \in X} s(i), \text{ and } \operatorname{argmin}_X(s) \coloneqq \{i \in X \mid s(i) = \min_X(s)\}.$$

We define $\max_X(s)$ and $\operatorname{argmax}_X(s)$ accordingly. 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 a and b be two integer sequences of the same length n.

(i) If for all $i \in [n]$, $a(i) \le b(i)$, then we write '$a \le b$'.

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

Definition 2.2 (Extensions). Let a be a sequence of length n. We define the set $E(a)$ of *extensions* of a as the set of sequences that are obtained from a by repeating each of its elements an arbitrary number of times. Formally, we let $E(a) \coloneqq \{a^* \mid \exists t_1, \ldots, t_n \colon \forall i \in [n], \forall j \in [t_i..(t_{i+1} - 1)] \colon a^*(j) = a(i)\}$.

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

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

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

2.1 Typical Sequences

We now define typical sequences and restate several lemmas due to Bodlaender and Kloks [8] that will be used throughout this text.

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

Removal of equal consecutive elements. If there is an index $i \in [n-1]$ such that $a(i) = a(i+1)$, then we change the sequence a from $a(1), \dots, a(i), a(i+1), \dots, a(n)$ to $a(1), \dots, a(i), a(i+2), \dots, a(n)$.

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

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

Lemma 2.7 (Bodlaender and Kloks [8]). *Let a and b be two integer sequences.*

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

(ii) (Lem. 3.13 in [8]). Suppose a and b are of the same length and let $y = a + b$. Let $a_0 \prec a$ and $b_0 \prec b$. Then there is an integer sequence $y_0 \in a_0 \oplus b_0$ such that $y_0 \prec y$.

(iii) (Lem. 3.14 in [8]). Let $c \in a \oplus b$. Then, there is an integer sequence $c' \in \tau(a) \oplus \tau(b)$ such that $c' \prec c$.

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

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

We may view a typical sequence $\tau(s)$ of an integer sequence s as a subsequence of s. Note that while $\tau(s)$ is unique, the choice of indices that induce $\tau(s)$ may not be unique. Our goal now is to show that given an integer sequence, we can compute its typical sequence in linear time. The following proposition captures the main property that will be used in our algorithm.

Proposition 2.8. *Let s be an integer sequence and let $i_{\min} := \operatorname{argmin}^\star(s)$. Let $1 =: j_0 < j_1 < j_2 < \dots < j_t < j_{t+1} := i_{\min}$ be pairwise distinct integers. 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 $[i_{\min}]$ 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 $\operatorname{argmin}^\star$ and $\operatorname{argmax}^\star$,

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

We prove the following statement. Under the stated conditions, for a given $h \in [0..t+1]$, the typical sequence of s restricted to $[j_h..i_{\min}]$ 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 element is always part of the typical sequence, and since $[j_{t+1}..i_{\min}] = \{i_{\min}\}$.

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 $[j_{h+1}..i_{\min}]$ 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 $s(i) \leq s(j_h) \leq s(i')$, or $s(i) \geq s(j_h) \geq s(i')$, 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 $s(j_h) = \operatorname{argmax}(s[1..j_{h+1}])$. 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 $[j_{h+1}..i_{\min}]$. 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 3.12.

Corollary 2.9. Let t be a typical sequence of length n such that $n \in \operatorname{argmin}(t)$. Then, for each $k \in \left[\lfloor \frac{n}{2} \rfloor\right]$, $n - 2k + 1 \in \operatorname{argmax}_{[n-(2k+1)]}(t)$ and $n - 2k \in \operatorname{argmin}_{[n-2k]}(t)$.

Equipped with Proposition 2.8, we can now proceed and give the linear-time algorithm that computes a typical sequence of an integer sequence.

Lemma 2.10. *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 $s(i)$. The remainder of the algorithm is closely inspired by Proposition 2.8. Throughout the following, we assume that $\mathrm{argmin}^\star(s) < \mathrm{argmax}^\star(s)$. (If not, then simply apply the following algorithm on the reverse sequence $s(n)$, $s(n-1)$, …, $s(1)$ and return the reverse of its output.) We first find $i_{\min} := \mathrm{argmin}^\star(s)$, and then we mark a set of indices in $[1..i_{\min}]$ that satisfy the preconditions of Proposition 2.8. This allows us to conclude that these indices induce $\tau(s)$ on $[1..i_{\min}]$. Next, letting $i_{\max}$ be the *rightmost* occurrence[2] of the maximum element in s, we apply the symmetric procedure on the sequence $s(n)$, $s(n-1)$, …, $s(i_{\max}+1)$, $s(i_{\max})$ which gives the remaining indices of the typical sequence of s. We focus on the description of the algorithm that finds the elements of the typical sequence in $[1..i_{\min}]$.

```
1 $j_{\min} \leftarrow \mathrm{argmin}^\star(s[1..2])$, $j_{\max} \leftarrow \mathrm{argmax}^\star(s[1..2])$, $M \leftarrow \{1\}$
2 for $j = 3, \ldots, i_{\min}$ do
3     if $s(j) < s(j_{\min})$ then
4         $j_{\min} \leftarrow j$
5         $M \leftarrow M \cup \{j_{\max}\}$ // mark the current value of $j_{\max}$
6     if $s(j) > s(j_{\max})$ then
7         $j_{\max} \leftarrow j$
8         $M \leftarrow M \cup \{j_{\min}\}$ // mark the current value of $j_{\min}$
9 $M \leftarrow M \cup \{j_{\min}\}$
```

Algorithm 1: The algorithm of Lemma 2.10 that computes the set M of indices that induce the typical sequence of s between the first element and the minimum element of s.

We execute Algorithm 1, which processes the integer sequence $s[1..i_{\min}]$ 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. It also keeps a set of *marked* indices M which will store the indices of the elements of the typical sequence. We now argue the correctness via Proposition 2.8.

Claim 2.10.1. The set M of indices marked by the above procedure induce $\tau(s)$ on $[1..i_{\min}]$.

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 2.8. First, we observe that the above algorithm marks both index 1 and index $i_{\min}$, in particular that $j_0 = 1$ and $j_{t+1} = i_{\min}$.

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.

We also 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 2.8. Suppose for a contradiction that for some $h \in [0..t+1]$, j_h violates this property. Assume that $s(j_h) < s(j_{h+1})$

[2] I.e. the occurrence with the largest index.

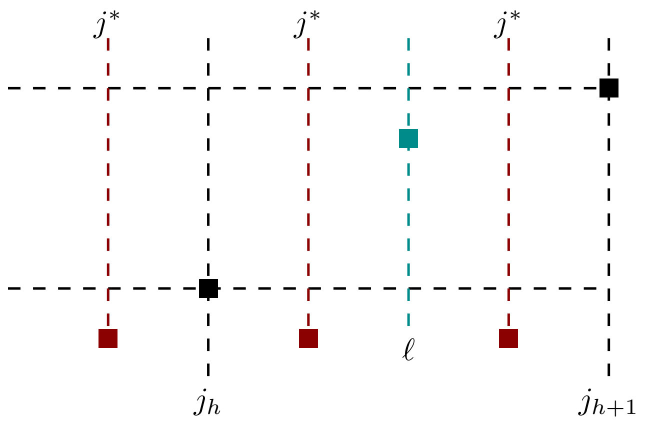


Figure 1: Illustration of the final argument in the proof of Claim 2.10.1. We assume that $s(j_h) < s(j_{h+1})$, and mark the possible positions for $j^* = \text{argmin}^\star(s[1..j_{h+1}])$ with $j^* \neq j_h$.

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 1. 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}. $\diamond$

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. $\square$

2.2 Directed Acyclic Graphs

A *directed graph* (or *digraph*) G is a pair of a set of *vertices* $V(G)$ and an ordered set of *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 $\mathsf{cutw}(\pi) := \max_{i \in [n-1]} |\{uv \in A(G) \mid \pi(u) \le i \wedge \pi(v) > i\}|$.

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

(iii) The *vertex separation number* of π is

$$\mathsf{vsn}(\pi) := \max_{i \in [n]} |\{u \in V(G) \mid \exists v \in V(G) \colon uv \in A(G) \wedge \pi(u) \le i \wedge \pi(v) > i\}|.$$

We define the cutwidth, modified cutwidth, and vertex separation number 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 [23].

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 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 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. [23] 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. [23]). *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 3.9 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 show that in M_1, we can find a dominating path by repeatedly 'chopping away' the last two rows or columns of M_1 and the first two rows or columns of M_2 and remembering a vertical or horizontal length-2 path in each step and case. 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.

We will in fact prove the Merge Dominator Lemma in terms of a more strict notion of domination which we call *strong* domination. This is not necessary to prove the lemma, however we will need the result in this stronger form for one of the applications presented in Section 4.

3.1 The Merge Matrix, Paths, and Strong Domination

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 $p(1), \ldots, p(r)$ of indices from M such that

(i) $p(1) = (1,1)$ and $p(r) = (m,n)$, and

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

We denote by $\mathcal{P}(M)$ the set of all paths in M. A sequence $p(1), \ldots, p(r)$ 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 $M[q]$.[3] We also may 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 $t \in [r-1]$, let $p(t) = (i,j)$; then, $p(t+1) \in \{(i+1,j), (i,j+1)\}$.

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 $M[e]$. We introduce the notion of *strong domination* that only applies to pairs of *merges* of integer sequences, specifically pairs of paths in a merge matrix. This notion is of importance to one of the algorithmic applications presented in Section 4.

[3]Recall that by (1) on page 4, for a (partial) path p in a matrix M, $M[p] = M[p(1)], M[p(2)], \ldots, M[p(l(p))]$.

Definition 3.4 (Strong Domination Property, '$\ltimes$'). Let M be an integer matrix and let p and q be two (partial) paths in M. Let $e \in E(p)$ and $f \in E(q)$ such that e and f have the same length ℓ. We say that e *has the strong domination property over* f, in symbols '$e \ltimes f$' if the following holds. For each $k \in [\ell]$, let $e(k) = (i_p, j_p)$ and $f(k) = (i_q, j_q)$. Then, $i_p \leq i_q$, or $j_p \leq j_q$, or both.

Definition 3.5 (Strong Domination, '$\prec_\ltimes$'). Let M be an integer matrix and let $p, q \in \mathcal{P}(M)$. We say that p *strongly dominates* q if there are extensions e of p and f of q of the same length such that the following hold.

(i) $M[e] \leq M[f]$.

(ii) e has the strong domination property over f, i.e. $e \ltimes f$.

If p strongly dominates q, then we write $p \prec_\ltimes q$. If additionally, q also strongly dominates p, we write $p \equiv_\ltimes q$ and say that p and q are *strongly equivalent*. If p strongly dominates all paths in $\mathcal{P}(M)$, we write $p \prec_\ltimes \mathcal{P}(M)$.

Note that the relation '$\prec_\ltimes$' is transitive as well and that Lemma 2.7(ii) and (v) hold for strong domination as well. A consequence of Lemma 2.7(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 matrix M of integer sequences r and c, $\mathcal{P}(M)$ contains all merges of r and c of length at most $l(r) + l(c) - 1$.

Corollary 3.6. 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 a and b, we denote by $a \boxplus b$ the set of all *non-diagonal merges* of a and b, which are not allowed to have 'diagonal' steps: we have that for all $t \in a \boxplus b$ and all $i \in [l(t) - 1]$, if $t(i) = a(i_a) + b(i_b)$, then $t(i+1) \in \{a(i_a + 1) + b(i_b), a(i_a) + a(i_b + 1)\}$. As each non-diagonal merge directly corresponds to a non-diagonal path in the merge matrix (and vice versa), we naturally have a notion of strong domination of non-diagonal merges as well. Furthermore, we can consider a non-diagonal path in a merge matrix to be a non-diagonal merge and vice versa. The next lemma will allow us to conclude that all results that we prove in this section for (not necessarily non-diagonal) merges hold for non-diagonal merges as well.

Lemma 3.7. *Let a and b be two integer sequences of length m and n, respectively. For any merge $c \in a \oplus b$, there is a non-diagonal merge $c' \in a \boxplus b$ such that $c' \equiv_\ltimes c$. Furthermore, given c, c' can be found in time $\mathcal{O}(m + n)$.*

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

$$\min\{a(i_a) + b(i_b + 1), a(i_a + 1) + b(i_b)\} \leq \max\{a(i_a) + b(i_b), a(i_a + 1), b(i_b + 1)\},$$

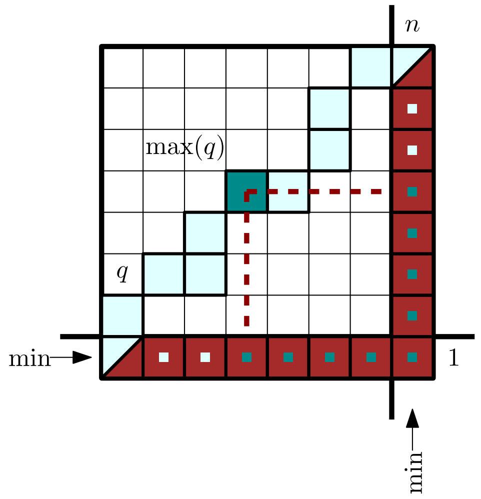


Figure 2: Situation in the proof of Lemma 3.8(i). The dot within each element of the corner path $p_\lrcorner$ indicates with which elements of the path q it is 'matched up' in the extensions constructed in the proof.

we have that the resulting sequence remains (strongly) equivalent to c. We let c' be the sequence obtained from c by applying this operation to all diagonal steps. It is clear that this can be implemented to run in time $\mathcal{O}(m+n)$. □

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:

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

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.

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 $j \in \operatorname{argmin}(c)$.*

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

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

Proof. (i) For an illustration of this proof see Figure 2. 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 $p_\lrcorner$ strongly dominates q.

- We let $p_\lrcorner^1 := p_\lrcorner(1),\ldots,p_\lrcorner(t_c^*-1)$ and $q_1 := q(1),\ldots,q(t^*-1)$.

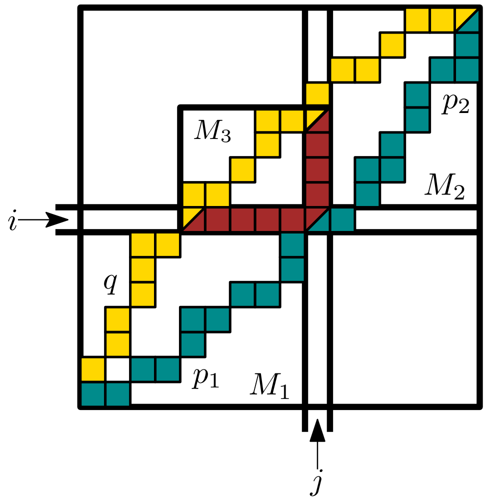


Figure 3: Situation in the proof of Lemma 3.9.

- We let $p^2_\lrcorner := p_\lrcorner(t^*_c), \dots, p_\lrcorner(n + t^*_r - 1)$ and $q_2 := q(t^*)$.
- We let $p^3_\lrcorner := p_\lrcorner(n + t^*_r), \dots, p_\lrcorner(m + n - 1)$ and $q_3 := q(t^* + 1), \dots, q(l(q))$.

Since $r(1)$ is a minimum row in M, we have that for all $(k, \ell) \in [m] \times [n]$, $M[1, \ell] \leq M[k, \ell]$. This implies that there is an extension e_1 of $p^1_\lrcorner$ of length $t^* - 1$ such that $M[e_1] \leq M[q_1]$. Furthermore, in this extension e_1, we have that for all $t \in [t^* - 1]$, $e_1(t)$ and $q_1(t)$ are from the same column in M, hence e_1 has the strong domination property over q_1. Similarly, there is an extension e_3 of $p^3_\lrcorner$ of length $l(q) - t^*$ such that $M[e_3] \leq M[q_3]$ and $e_3 \ltimes q_3$. 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 $p^2_\lrcorner \leq f_2$. It is clear that $p^2_\lrcorner \ltimes f_2$.

We define an extension e of $p_\lrcorner$ as $e := e_1 \circ p^2_\lrcorner \circ e_3$ and an extension f of q as $f := q_1 \circ f_2 \circ q_3$. Note that $l(e) = l(f) = l(q) + n + t^*_r - (t^*_c + 1)$. By the above argument we have that $M[e] \leq M[f]$, and that $e \ltimes f$, which finishes the proof. (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 element in the top right corner, and one the has the minimum element in the bottom left corner, compute a (strong) dominating path for each of them, and paste them together to obtain a (strong) dominating path for M.

Lemma 3.9 (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 $t \in [2]$, let $p_t \in \mathcal{P}(M_t)$ be a strong dominating path in M_t, i.e. $p_t \prec_\ltimes \mathcal{P}(M_t)$. Then, $p_1 \circ p_2$ is a strong dominating path in M, i.e. $p_1 \circ p_2 \prec_\ltimes \mathcal{P}(M)$.*

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, $p_1 \prec_{\ltimes} q_1$ and $p_2 \prec_{\ltimes} q_2$, and for $i \in [2]$, there are extensions e_i of p_i and f_i of q_i of the same length such that $M[e_i] \leq M[f_i]$ and $e_i \ltimes f_i$. We can conclude that $M[e_1] \circ M[e_2] \leq M[f_1] \circ M[f_2]$ and $e_1 \circ e_2 \ltimes f_1 \circ f_2$ which implies that $p \prec_{\ltimes} q$.

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 3.

Claim 3.9.1. Let p and q be as above, and suppose that q contains (i,j'), where $j' < j$. Then, $p \prec_{\ltimes} q$.

Proof. In this case, q also contains some (i',j), where $i' > i$. Let j'_i be the index of (i,j') in q, i.e. $q(j'_i) = (i,j')$, and i'_j denote the index of (i',j) in q, i.e. $q(i'_j) = (i',j)$. We derive the following sequences from q.

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

Since $q_1^+ \in \mathcal{P}(M_1)$ and $p_1 \prec_{\ltimes} \mathcal{P}(M_1)$, we have that $p_1 \prec_{\ltimes} q_1^+$, similarly that $p_2 \prec_{\ltimes} q_2^+$ and considering $M_3 := M[i'..i, j..j']$, we have by Lemma 3.8(i) that $p_{12} := p_{\lrcorner}(M_3) = (i,j'),(i,j'+1),\ldots,(i,j),(i+1,j),\ldots,(i',j)$ strongly 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 $l(e_1) = l(f_1)$, $M[e_1] \leq M[f_1]$ and $e_1 \ltimes f_1$.

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

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

We construct extensions $e' \in E(p)$ and $f' \in E(q)$ as follows. First, let a be the index of the last repetition in f_1 of $q(j'_i - 1)$, i.e. the index that appears just before $q(j'_i) = (i,j')$ in q. We let $e'_{j'-1}[1..a] := e_1[1..a]$ and $f'_{j'-1}[1..a] := f_1[1..a]$. By (I), $M[e'_{j'-1}] \leq M[f'_{j'-1}]$ and $e'_{j'-1} \ltimes f'_{j'-1}$.

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 4. We maintain as an invariant that $l(e'_{x-1}) = l(f'_{x-1})$ and that $M[e'_{x-1}] \leq M[f'_{x-1}]$ and $e'_{x-1} \ltimes f'_{x-1}$. 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 $y' \in \{b_1, \ldots, b_d\}$,

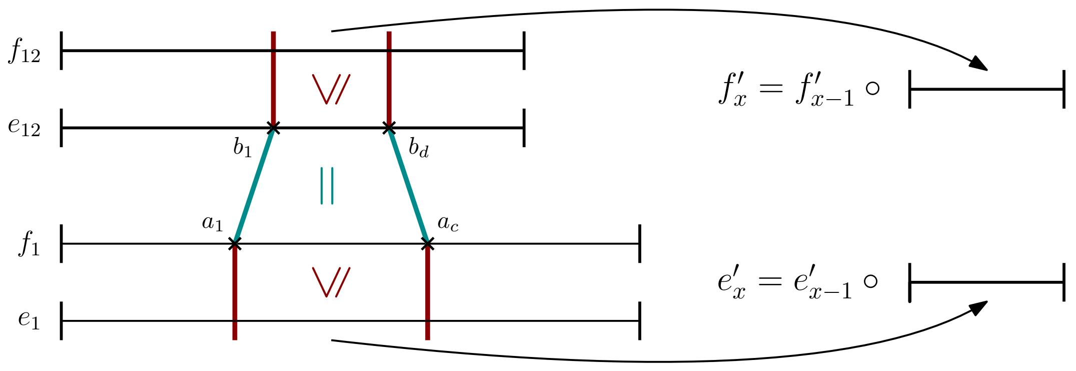


Figure 4: Constructing extensions in the proof of Claim 3.9.1.

$f_1(y) = e_{12}(y')$ and we know that $M[e_{12}(y')] \le M[f_{12}(y')]$ by (II). Hence, $M[e'_x] \le M[f'_x]$ in either of the above cases. Furthermore, one can verify that $e'_x \ltimes f'_x$.

We proceed analogously to construct, say, $e'_{j+i}, \dots, e'_{j+i'}$ and $f'_{j+i}, \dots, f'_{j+i'}$ using (III) and obtain e' from $e'_{j+i'}$ and f' from $f_{j+i'}$, appending the corresponding parts from e_2 and f_2, respectively. The latter step can be done symmetrically to how we constructed $e_{j'-1}$ and $f_{j'-1}$. This completes the proof. $\diamond$

The proof of this claim finishes the proof of Lemma 3.9. $\square$

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 will yield a (strong) 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.

Lemma 3.10. *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 $k \in \operatorname{argmin}_{[m]\setminus\{i\}}(r)$ and $\ell \in \operatorname{argmin}_{[n]\setminus\{j\}}(c)$. Let $\{p^*, q^*\} = \{p_\lrcorner, p_\ulcorner\}$ such that $\max(M[p^*]) \le \max(M[q^*])$.*

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

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

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 3.8. We prove the lemma in two steps:

1. We show that for each path q in M, $p_\lrcorner$ or $p_\ulcorner$ (or both) strongly dominate(s) q.
2. We show that $p_\lrcorner$ strongly dominates $p_\ulcorner$, or vice versa, or both.

The following claim will be useful in both steps and can be seen as a slight generalization of Lemma 3.8.

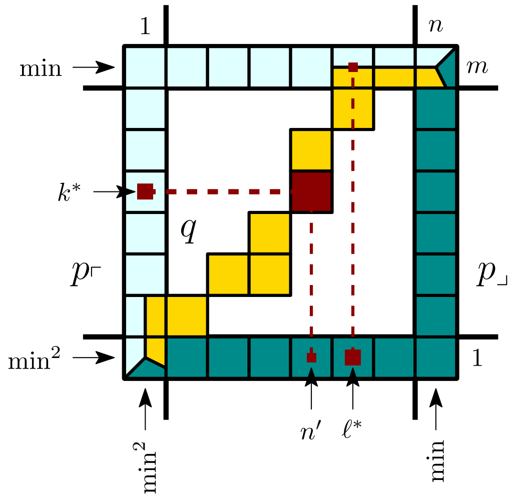


Figure 5: Situation in the first stage of the proof of Lemma 3.10(i). The row and column labeled 'min' contains the minimum element from the respective sequence, and the row and column labeled '$\min^2$' contains the minimum element among all elements except the one in the min-row or -column.

Claim 3.10.1. Let $q \in \mathcal{P}(M)$ and let $p \in \{p_\lrcorner, p_\ulcorner\}$. If $\max(M[p]) \leq \max(M[q])$, *then* $p \prec_\ltimes q$.

Proof. Suppose (wlog) that $p = p_\lrcorner$. The claim can be shown using the same argument as in Lemma 3.8, 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. ◇

We prove Step 1. For the following argument, see Figure 5. If $\max(M[q]) \geq \max(M[p_\lrcorner])$, then we conclude by Claim 3.10.1 that $p_\lrcorner \prec_\ltimes q$ and we are done with Step 1 of the proof. Suppose

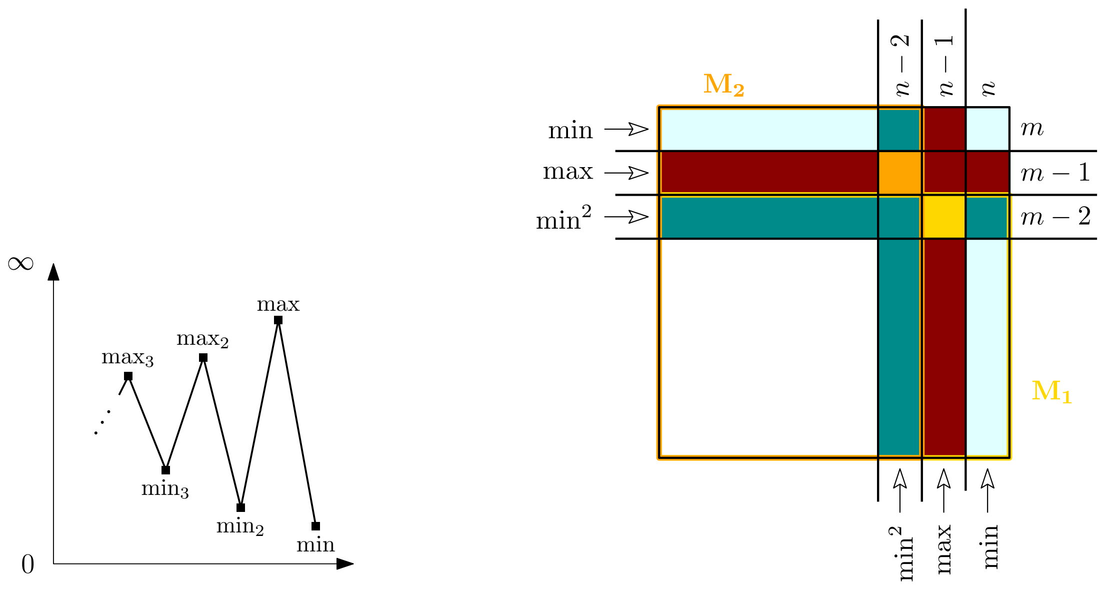
$$\max(M[q]) < \max(M[p_\lrcorner]) \tag{3}$$

and let $\ell^* := \operatorname{argmax}(M[p_\lrcorner])$. We may assume that $\ell^* < n$. We furthermore have that q contains (m, ℓ^*), as this is the only way in which (3) can be satisfied. Now let $k^* := \operatorname{argmax}(M[p_\ulcorner])$. We may assume that $k^* < m$. Now, since q contains (m, ℓ^*), we have that q also contains some (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 3.10.1, $p_\ulcorner$ strongly dominates q and we finished Step 1 of the proof. Step 2 follows from another application of Claim 3.10.1 and the lemma follows from transitivity of the (strong) domination relation. This proves (i), and (ii) follows from a symmetric argument. □

Remark 3.11. 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.



(a) Typical sequence ending in the minimum value.

(b) The merge matrix and submatrices considered in the proof of Lemma 3.12.

Figure 6: Visual aides to the proof of Lemma 3.12.

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.

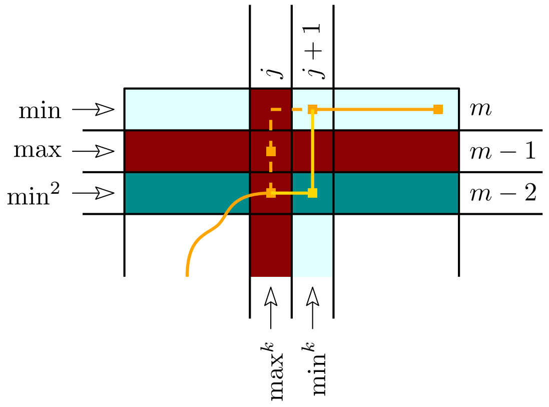
Lemma 3.12 (Chop Lemma - Bottom). *Let r and c be typical sequences of length $m \geq 3$ and $n \geq 3$, respectively, and let M be the merge matrix of r and c. Suppose that $m \in \operatorname{argmin}(r)$ and $n \in \operatorname{argmin}(c)$ and let $M_1 := M[1..(m-2), 1..n]$ and $M_2 := M[1..m, 1..(n-2)]$ and for $t \in [2]$, let $p_t \prec_{\ltimes} \mathcal{P}(M_t)$.*

(i) If $M[m-2, n-1] \leq M[m-1, n-2]$, then $p_1^+ := p_1 \circ (m-1, n), (m, n)$ strongly dominates $\mathcal{P}(M)$, i.e. $p_1^+ \prec_{\ltimes} \mathcal{P}(M)$.

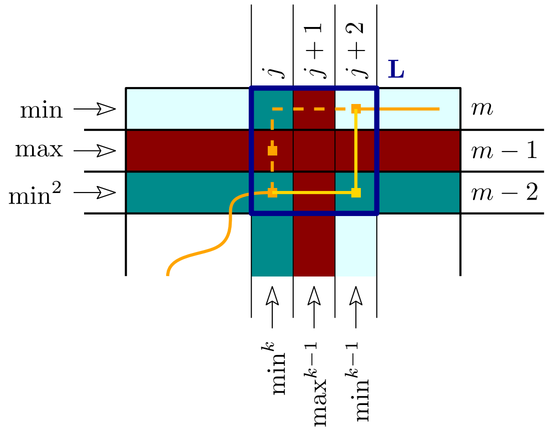
(ii) If $M[m-1, n-2] \leq M[m-2, n-1]$, then $p_2^+ := p_2 \circ (m, n-1), (m, n)$ strongly dominates $\mathcal{P}(M)$, i.e. $p_2^+ \prec_{\ltimes} \mathcal{P}(M)$.

Proof. Let $s \in \{r, c\}$. Since s is a typical sequence and $l(s) \in \operatorname{argmin}(s)$, we know by Corollary 2.9 that for all $k \in \left[\left\lfloor \frac{l(s)}{2} \right\rfloor\right]$, $n - 2k + 1 \in \operatorname{argmax}_{[n-2k+1]}(t)$ and $n - 2k \in \operatorname{argmin}_{[n-2k]}(t)$. 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 $n-2k$ ($2k \leq n$) as '$\min^{k+1}(s)$' (note that the minimum is achieved when $k = 0$, hence the '+1'), and elements at position $n - 2k + 1$ ($2k + 1 \leq n - 1$) as '$\max^k(s)$'. For an illustration of the shape of s see Figure 6a and for an illustration of the basic setting of this proof see Figure 6b. We prove (i) and remark that the argument for (ii) is symmetric.

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



(a) Situation of Claim 3.12.3.



(b) Situation of Claim 3.12.4.

Figure 7: Visualization of the arguments that lead to the conclusion that p_2 contains $(m-2, n-2)$ in the proof of Lemma 3.12.

Claim 3.12.1. Let $q \in \mathcal{P}(M)$. Then, for some $r \in [2]$, $p_r^+ \prec_\ltimes q$.

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' strongly 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 strongly 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 strongly dominates all paths in M_1, it strongly dominates $q|_{M_1}$ and so $p_1^+ \prec_\ltimes q$. ◇

The remainder of the proof is devoted to showing that p_1^+ strongly dominates p_2^+ which yields the lemma by Claim 3.12.1 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 strongly dominates p_2.

Claim 3.12.2. 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 $j_1, \ldots, j_r$ be such that p_2 contains all $(m-1, j_i)$, where $i \in [r]$. By the definition of a path in a matrix, we have that $j_{i+1} = j_i + 1$ for all $i \in [r-1]$. Let p_2' be the path obtained from p_2 by replacing, for each $i \in [r-1]$, 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' strongly dominates p_2, and clearly, p_2' satisfies the condition of the claim. ◇

Claim 3.12.3. 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 strongly dominates p_2 and contains $(m-1, j+1)$.

Proof. For an illustration see Figure 7a. First, by Claim 3.12.2, 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 strong dominating path since $M[m, j] > M[m, j+1]$. We modify p_2 as follows. We remove $(m - 1, j)$, and add $(m - 2, j)$ (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 $c(j + 1) < c(j)$), and furthermore, for each such x, if (i_x, j_x) is the index of x in M, we have that $i_x \le m - 1$. Hence, the resulting path strongly dominates p_2 and it contains $(m - 1, j + 1)$. ◇

Claim 3.12.4. 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 strongly dominates p_2 and contains $(m - 1, j + 2)$.

Proof. For an illustration see Figure 7b. Again, by Claim 3.12.2, 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 (strong) 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 $L \coloneqq M[(m-2)..m, j..(j+2)]$. 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 $p_{\lrcorner}(L)$ strongly 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 strongly dominates p_2 by replacing $p_{\ulcorner}(L)$ with $p_{\lrcorner}(L)$. By Lemma 3.10, it suffices to show that $M[m - 2, j + 1] \le M[m - 1, j]$, in other words, that $\max^{k-1}(c) + \min^2(r) \le \max(r) + \min^k(c)$.

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

$$\max(c) + \min^2(r) \le \max(r) + \min^2(c), \text{ and so: } \max(c) - \min^2(c) \le \max(r) - \min^2(r).$$

Next, we have that for all $j \in \left[\left\lfloor \frac{n}{2} \right\rfloor\right]$,

$$\max(c) - \min^2(c) \ge \max^j(c) - \min^{j+1}(c).$$

Putting the two together, we have that

$$\max^{k-1}(c) - \min^k(c) \le \max(r) - \min^2(r), \text{ and so: } \max^{k-1}(c) + \min^2(r) \le \max(r) + \min^k(c),$$

which concludes the proof of the claim. ◇

After exhaustive applications of Claims 3.12.3 and 3.12.4, we may assume that p_2 contains $(m - 1, n - 2)$. We can conclude that we may assume that p_2 contains $(m - 2, n - 2)$. (Simply add this element if it is not already present.) We are now ready to conclude the proof.

Claim 3.12.5. $p_1^+ \prec_{\ltimes} p_2^+$.

Proof. Let L be the 3×3 submatrix $M[(m-2)..m, (n-2)..n]$. By the above claims, we may assume that p_2^+ restricted to L is precisely $p_{\ulcorner}(L)$. We obtain p_2^- from p_2^+ by replacing that part with $p_{\lrcorner}(L)$. Clearly, p_1^+ strongly dominates p_2^- which in turn strongly dominates p_2^+: the latter can be seen by an application of Lemma 3.10 to L. ◇

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 $n \leq 2$.

Observation 3.13 (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[4]

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

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

By symmetry, we have the following consequence of Lemma 3.12.

Corollary 3.14 (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 $M_1 := M[3..m, 1..n]$ and $M_2 := M[1..m, 3..n]$ and for $t \in [2]$, let $p_t \prec_{\ltimes} \mathcal{P}(M_t)$.*

(i) If $M[3,2] \leq M[2,3]$, then $p_1^+ := (1,1),(2,1) \circ p_1 \prec_{\ltimes} \mathcal{P}(M)$.

(ii) If $M[2,3] \leq M[3,2]$, then $p_2^+ := (1,1),(1,2) \circ p_2 \prec_{\ltimes} \mathcal{P}(M)$.

Again, we observe the corresponding base case.

Observation 3.15 (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

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

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

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 3.16. *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 strongly dominating path in the merge matrix of r and c.*

Proof. The algorithm practically derives itself from the Split Lemma (Lemma 3.9) and the Chop Lemmas (Lemma 3.12 and Corollary 3.14). 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 3.9 and 3.12 and Corollary 3.14 with the base cases given in Observations 3.13 and 3.15. ☐

[4] 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.

```
    Input  : Typical sequences $r(1),\dots,r(m)$ and $c(1),\dots,c(n)$
    Output: A strong dominating merge of $r$ and $c$
1   Let $i \in \operatorname{argmin}(r)$ and $j \in \operatorname{argmin}(c)$;
2   return Chop-bottom $(r[1..i], c[1..j]) \circ$ Chop-top $(r[i..m], c[j..n])$;
3   Procedure Chop-bottom($r$ and $c$ as above)
4       if $m \leq 2$ then return $r(1)+c(1)$, $r(m)+c(1)$, $r(m)+c(2)$, $\dots$, $r(m)+c(n)$;
5       if $n \leq 2$ then return $r(1)+c(1)$, $r(1)+c(n)$, $r(2)+c(n)$, $\dots$, $r(m)+c(n)$;
6       if $r(m-2)+c(n-1) \leq r(m-1)+c(n-2)$ then return
          Chop-bottom($r[1..(m-2)], c$) $\circ\, r(m-1)+c(n), r(m)+c(n)$;
7       if $r(m-1)+c(n-2) \leq r(m-2)+c(n-1)$ then return
          Chop-bottom($r, c[1..(n-2)]$) $\circ\, r(m)+c(n-1), r(m)+c(n)$;
8   Procedure Chop-top($r$ and $c$ as above)
9       if $m \leq 2$ then return $r(1)+c(1)$, $r(1)+c(2)$, $\dots$, $r(1)+c(n)$, $r(m)+c(n)$;
10      if $n \leq 2$ then return $r(1)+c(1)$, $r(2)+c(1)$, $\dots$, $r(m)+c(1)$, $r(m)+c(n)$;
11      if $r(3)+c(2) \leq r(2)+c(3)$ then return $r(1)+c(1), r(2)+c(1) \circ$ Chop-top($r[3..m], c$);
12      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

3.5 Generalization to Arbitrary Integer Sequences

In this section we show how to generalize Lemma 3.16 to arbitrary integer sequences. In particular, we will show how to construct from a merge of two typical sequences $\tau(a)$ and $\tau(b)$ that dominates all of their merges, an merge of a and b that dominates all merges of a and b. The claimed result then follows from an application of Lemma 3.16.

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

Step 1. We construct $t' \in \tau(a) \boxplus \tau(b)$ such that $t' \equiv_{\ltimes} t$ using Lemma 3.7. Throughout the following, consider t' to be a path in the merge matrix of $\tau(a)$ and $\tau(b)$.

Step 2. First, we initialize $\rho_t^1 \coloneqq t'(1) = (1,1)$. For $i = \{2,\dots,l(t')\}$, we proceed inductively as follows. Let $(i_a, i_b) = t(i)$ and let $(i'_a, i'_b) = t(i-1)$. Let furthermore (j_a, j_b) be the index in M corresponding to (i_a, i_b), and let (j'_a, j'_b) be the index in M corresponding to (i'_a, i'_b). Assume by induction that $\rho_t^{i-1} \in \mathcal{P}(M[1..j'_a, 1..j'_b])$. We show how to extend ρ_t^{i-1} to a path in ρ_t^i in $M[1..j_a, 1..j_b]$. Since t' is non-diagonal, we have that $(i'_a, i'_b) \in \{(i_a-1, i_b), (i_a, i_b-1)\}$, so one of the two following cases applies.

Case S2.1 ($i'_a = i_a - 1$ and $i'_b = i_b$). In this case, we let $\rho_t^i \coloneqq \rho_t^{i-1} \circ (j'_a+1, j_b), \dots, (j_a, j_b)$.

Case S2.2 ($i'_a = i_a$ and $i'_b = i_b - 1$). In this case, we let $\rho_t^i \coloneqq \rho_t^{i-1} \circ (j_a, j'_b+1), \dots, (j_a, j_b)$.

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

We illustrate this construction in Figure 8. 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 $\tau(a)$

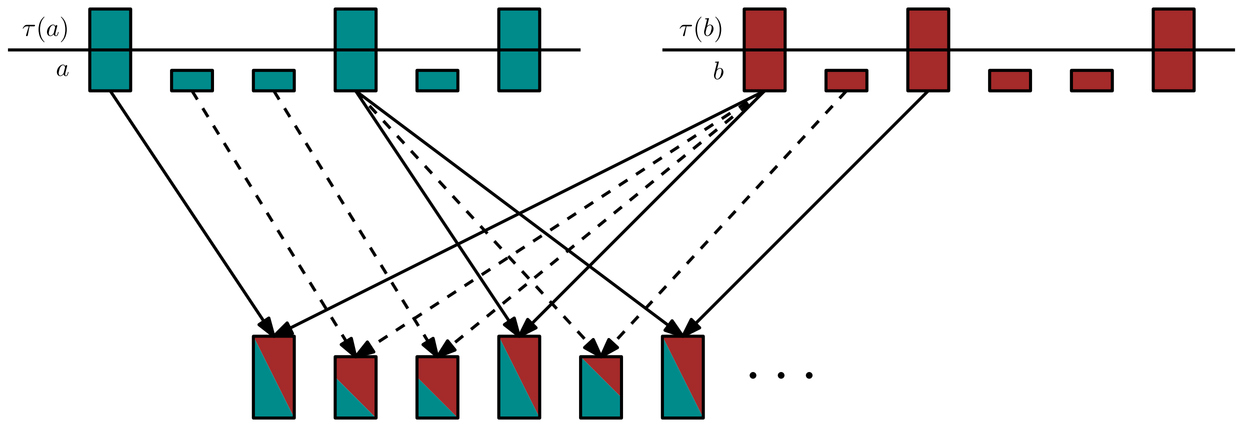


Figure 8: Illustration of the typical lift of some $t \in \tau(a) \oplus \tau(b)$. On the top, the large blocks on the top show elements of the typical sequences $\tau(a)$ and $\tau(b)$, while the small blocks show elements of a and b that are not part of the respective typical sequence. The large blocks on the bottom correspond to elements of some $t' \in \tau(a) \boxplus \tau(b)$ with $t' \equiv_{\ltimes} t$ as constructed in Step 1, with the arrows indicating which elements from $\tau(a)$ and $\tau(b)$ are being added together to obtain them. The smaller blocks on the bottom show elements of $\rho(t)$ with the dashed arrows pointing to which elements from a and b are being added together to obtain them.

and $\tau(b)$ by inserting vertical and horizontal paths from the merge matrix of a and b between consecutive elements. Moreover, it is computable in linear time, with Step 1 taking linear time by Lemma 3.7. We summarize in the following observation.

Observation 3.17. Let a and b be integer sequences of length m and n, respectively, and let $t \in \tau(a) \oplus \tau(b)$ be such that $t \prec_{\ltimes} \tau(a) \oplus \tau(b)$. Then, $\rho(t) \in a \boxplus b$, and $\rho(t)$ can be computed in time $\mathcal{O}(m + n)$.

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

Lemma 3.18. *Let a and b be integer sequences and let $c \in a \oplus b$. Let $t \in \tau(a) \oplus \tau(b)$ such that $t \prec_{\ltimes} \tau(a) \oplus \tau(b)$. Then, $\rho(t) \prec_{\ltimes} c$.*

Proof. First, it is not difficult to verify that $\rho(t) \equiv t$. Let t' be the non-diagonal merge that is strongly equivalent to t as used in Step 1 of the construction of $\rho(t)$. By Lemma 3.7, there exists a non-diagonal merge $c'' \in a \boxplus b$ such that $c'' \equiv_{\ltimes} c$. By Lemma 2.7(iii) and Lemma 3.7, there exists a $c' \in \tau(a) \boxplus \tau(b)$ such that $c' \prec c''$. Moreover, we may assume that the following holds. Let M be the merge matrix of a and b and let M_τ be the submatrix of M induced only by the rows and columns that correspond to elements of $\tau(a)$ and $\tau(b)$. Then, we may view $\rho(t)$ restricted to M_τ as t', and c'' restricted to M_τ as c'. We will show that $\rho(t) \prec_{\ltimes} c''$, and hence $\rho(t) \prec_{\ltimes} c$.

For the sake of our arguments, throughout the following, we will always view t', $\rho(t)$, c', and c as paths in the corresponding merge matrices. Since $t' \prec_{\ltimes} c'$, there are extensions $e_{t'} \in E(t')$ and $e_{c'} \in E(c')$ of the same length ℓ such that $M_\tau[e_{t'}] \leq M_\tau[e_{c'}]$ and $e_{t'} \ltimes e_{c'}$. Our goal is now to modify $e_{t'}$ and $e_{c'}$ such that these extensions have some properties (described below) that can be exploited to obtain a pair of extensions $e_{\rho(t)} \in E(\rho(t))$ and $e_{c''} \in E(c'')$ of the same length such that $M[e_{\rho(t)}] \leq M[e_{c''}]$ and $e_{\rho(t)} \ltimes e_{c''}$.

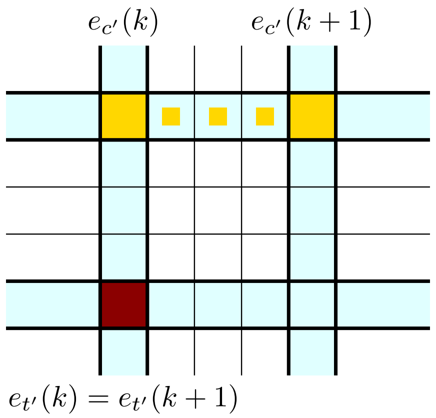


Figure 9: Illustration of the first part of the argument given in the proof of Lemma 3.18. A part of the merge matrix M is shown, and the colored rows and columns correspond to the ones in M_τ. The small dots between $e_{c'}(k)$ and $e_{c'}(k+1)$ show a partial path between said elements in M.

Throughout the following, we let $\mu_{c'} := M_\tau[e_{c'}]$ and $\mu_{t'} := M_\tau[e_{t'}]$. Assume for instance that for some $k \in [\ell-1]$, $e_{t'}(k) = e_{t'}(k+1)$, and $\mu_{c'}(k) \le \mu_{c'}(k+1)$. Then, $\mu_{t'}(k) \le \mu_{c'}(k) \le \mu_{c'}(k+1)$, and $e_{c'}(k)$, $e_{c'}(k+1)$ corresponds to a vertical or horizontal step in M_τ, and hence to a vertical or horizontal partial path in M. On that partial path, by the definition of a typical sequence, each element z is such that $\mu_{c'}(k) \le M[z] \le \mu_{c'}(k+1)$, implying that $\mu_{t'}(k) \le \mu_{c'}(k) \le M[z]$. Hence, we can insert the partial path between $e_{c'}(k)$ and $e_{c'}(k+1)$ in M between positions k and $k+1$ in $e_{c'}$, repeat $e_{t'}(k)$ the corresponding number of times, and we do not violate the property that $M[e_{t'}] \le M[e_{c'}]$. Moreover, as along that path, $e_{c'}(k)$ is the 'smallest' index in M and $e_{t'}(k) = e_{t'}(k+1)$, the insertion of the path does not cause a violation of the strong domination property either. The situation when $\mu_{c'}(k) > \mu_{c'}(k+1)$ is symmetric. For an illustration of this argument, see Figure 9.

If on the other hand, for some $k \in [\ell-1]$, $\mu_{t'}(k) \le \mu_{t'}(k+1)$ and $e_{c'}(k) = e_{c'}(k+1)$, then for all z on the partial (horizontal or vertical) path in M between $e_{t'}(k)$ and $e_{t'}(k+1)$ we have $\mu_{t'}(k) \le M[z] \le \mu_{t'}(k+1)$, and hence $M[z] \le \mu_{t'}(k+1) \le \mu_{c'}(k)$. Hence, we can insert said partial path between positions k and $k+1$ in $e_{t'}$, and repeat $e_{c'}(k)$ the corresponding number of times. The situation when $\mu_{t'}(k) > \mu_{t'}(k+1)$ is symmetric.

Hence, if for all $k \in [\ell-1]$, either $e_{t'}(k) = e_{t'}(k+1)$ or $e_{c'}(k) = e_{c'}(k+1)$, then we know how to modify $e_{t'}$ and $e_{c'}$ to obtain extensions of $\rho(t)$ and of c, respectively, that witness that $\rho(t) \prec_\ltimes c$. In the next claim we show that if for some $k \in [\ell-1]$, $e_{t'}(k) \neq e_{t'}(k+1)$ and $e_{c'}(k) \neq e_{c'}(k+1)$, then we can always locally manipulate these extensions so that they satisfy our desired condition, unless we are in a situation that we can exploit in a different way.

Throughout the following, for $k \in [\ell]$, we use the notation $e_{c'}(k) = (\iota_c(k), \eta_c(k))$ and $e_{t'}(k) = (\iota_t(k), \eta_t(k))$.

Claim 3.18.1. Let $e_{t'}$, $e_{c'}$ be as above. We may assume that for each $k \in [\ell-1]$, either

(i) $e_{t'}(k) = e_{t'}(k+1)$ or $e_{c'}(k) = e_{c'}(k+1)$, or

(ii) $\iota_t(k) = \iota_c(k) =: i$ and $\iota_t(k+1) = \iota_c(k+1) = i+1$, or $\eta_t(k) = \eta_c(k) =: j$ and $\eta_t(k+1) = \eta_c(k+1) = j+1$.

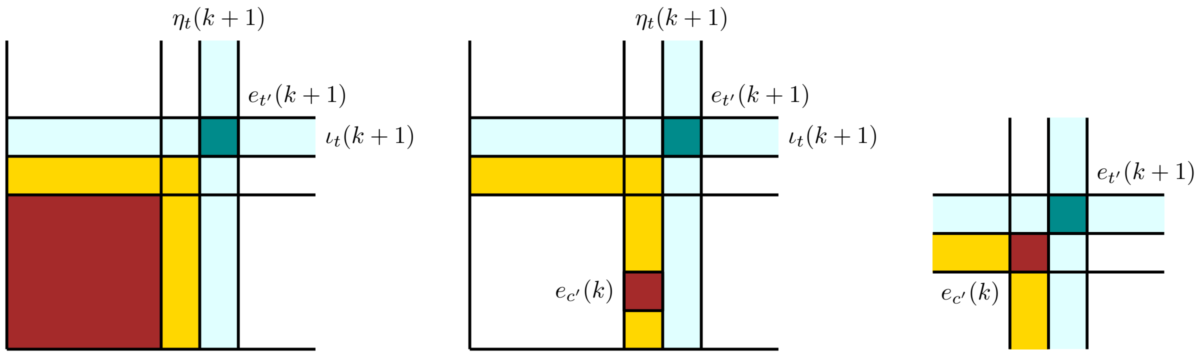


Figure 10: Illustrations of situations that occur in Case 1 of the proof of Claim 3.18.1.

Proof. Let $k \in [\ell - 1]$ be such that $e_{t'}(k) \neq e_{t'}(k+1)$ and $e_{c'}(k) \neq e_{c'}(k+1)$.

Case 1 ($\mu_{c'}(k) < \mu_{c'}(k+1)$). In this case, $\mu_{t'}(k) \leq \mu_{c'}(k) < \mu_{c'}(k+1)$; we insert a copy of $e_{c'}(k+1)$ between position k and $k+1$ in $e_{c'}$, and a copy of $e_{t'}(k)$ between position k and $k+1$ in $e_{t'}$, call the resulting extensions $e'_{t'}$ and $e'_{c'}$, respectively. We have to show that the fact that in the new extensions, $e_{t'}(k)$ appears at the same position as $e_{c'}(k+1)$ does not change the property that $M[e'_{t'}] \leq M[e'_{c'}]$ and $e'_{t'} \ltimes e'_{c'}$. As $\mu_{t'}(k) \leq \mu_{c'}(k+1)$, we immediately have that $M[e'_{t'}] \leq M[e'_{c'}]$.

Moreover, we know that $\iota_c(k+1) \geq \iota_c(k)$ and $\eta_c(k+1) \geq \eta_c(k)$, by the definition of a (partial) path in a matrix. Next, since $e_{t'} \ltimes e_{c'}$, we have that $\iota_t(k) \leq \iota_c(k)$, or $\eta_t(k) \leq \eta_c(k)$, or both. Suppose $\iota_t(k) \leq \iota_c(k)$. (The other case works the same way.) Then, $\iota_t(k) \leq \iota_c(k) \leq \iota_c(k+1)$, meaning $e'_{t'} \ltimes e'_{c'}$. Moreover, we have achieved that $e'_{t'}(k) = e'_{t'}(k+1)$ and $e'_{c'}(k+1) = e'_{c'}(k+2)$.

Case 2 ($\mu_{c'}(k) > \mu_{c'}(k+1)$). We now have that $\mu_{t'}(k+1) \leq \mu_{c'}(k+1) < \mu_{c'}(k)$. Analogously to the previous case, we may insert a copy of $e_{t'}(k+1)$ between position k and $k+1$ in $e_{t'}$, and a copy of $e_{c'}(k)$ between position k and $k+1$ in $e_{c'}$. However, in this case it might happen that we lose the strong domination property, namely when

$$\iota_t(k+1) > \iota_c(k) \text{ and } \eta_t(k+1) > \eta_c(k). \tag{4}$$

Suppose that (4) holds. We will derive that in this case, either we can apply the same modification as in Case 1, or we are in situation (ii). We first argue that we may assume that one of the following holds. For an illustration see Figure 10.

$$\iota_c(k) = \iota_t(k+1) - 1 \text{ and } \eta_c(k) \in [1..(\eta_t(k+1) - 1)] \tag{5}$$
$$\iota_c(k) \in [1..(\iota_t(k+1) - 1)] \text{ and } \eta_c(k) = \eta_t(k+1) - 1 \tag{6}$$

By (4), we know that $(\iota_c(k), \eta_c(k)) \in [\iota_t(k+1) - 1)] \times [\eta_t(k+1) - 1]$. Suppose additionally that $\iota_c(k) < \iota_t(k+1) - 1$ and $\eta_c(k) < \eta_t(k+1) - 1$. By the definition of a non-diagonal path, we know that $\iota_t(k) \geq \iota_t(k+1) - 1$ and $\eta_t(k) \geq \eta_t(k+1) - 1$. Hence, we have a contradiction with the fact that $e_{t'} \ltimes e_{c'}$, since under our assumptions, $\iota_t(k) \geq \iota_t(k+1) - 1 > \iota_c(k)$ and $\eta_t(k) \geq \eta_t(k+1) - 1 > \eta_c(k)$. There are three cases that remain to be considered.

Case 2.1 ($\iota_c(k) < \iota_t(k+1) - 1$). For an illustration of this case see the middle part of Figure 10. In this case, (6) holds, so we have that $\eta_c(k) = \eta_t(k+1) - 1$. Next, we observe that $\eta_t(k) = \eta_t(k+1) - 1$, as if $\eta_t(k) = \eta_t(k+1)$, then we have a contradiction with the fact that

$e_{t'} \ltimes e_{c'}$: in that case, $\eta_t(k) = \eta_t(k+1) > \eta_t(k+1) - 1 = \eta_c(k)$, and $\iota_t(k) = \iota_t(k+1) - 1 > \iota_c(k)$. Next, we argue that $\eta_c(k+1) = \eta_c(k)+1 = \eta_t(k+1)$. For if not, then $\eta_c(k+1) = \eta_c(k) = \eta_t(k+1)-1$, which yields a contradiction with $e_{t'} \ltimes e_{c'}$: we have $\eta_t(k+1) > \eta_t(k+1) - 1 = \eta_c(k+1)$, and $\iota_t(k+1) > \iota_c(k) + 1 \geq \iota_c(k+1)$.

We have shown that $\eta_t(k+1) = \eta_c(k+1)$ and $\eta_t(k+1) = \eta_t(k)+1$. (6) yields $\eta_c(k+1) = \eta_c(k)+1$, so $\eta_t(k) = \eta_c(k)$ and we are in situation (ii).

Case 2.2 ($\boldsymbol{\eta_c(k) < \eta_t(k+1) - 1}$). In this case, (5) holds, and an argument symmetric to the one given in the previous case reveals that $\iota_t(k) = \iota_c(k) =: i$ and $\iota_t(k+1) = \iota_c(k+1) = i+1$, so again we are in situation (ii).

Case 2.3 ($\boldsymbol{\iota_c(k) = \iota_t(k+1) - 1}$ and $\boldsymbol{\eta_c(k) = \eta_t(k+1) - 1}$). For an illustration of this case see the right side of Figure 10. If $e_{t'}(k) = e_{c'}(k+1)$, then the same modification as the one presented in Case 1 works: we can insert a copy of $e_{t'}(k+1)$ between position k and $k+1$ in $e_{t'}$, and a copy of $e_{c'}(k)$ in between position k and $k+1$ in $e_{c'}$, and we still have that $M[e_{t'}] \leq M[e_{c'}]$ and $e_{t'} \ltimes e_{c'}$. If $e_{t'}(k) \neq e_{c'}(k+1)$, then we are in situation (ii).

The end of this case analysis finishes the proof of the claim. $\diamond$

We can now finish the proof as follows. Assume that $e_{t'}$ and $e_{c'}$ have the properties of Claim 3.18.1. For each $k \in [\ell]$ satisfying situation (i), we have seen in the beginning of the proof how to modify $e_{t'}$ and $e_{c'}$ to accommodate for the elements on the partial path between $e_{t'}(k)$ and $e_{t'}(k+1)$ or between $e_{c'}(k)$ and $e_{c'}(k+1)$. Suppose the first case of situation (ii) applies, i.e. $\iota_t(k) = \iota_c(k) =: i$ and $\iota_t(k+1) = \iota_c(k+1) = i+1$. Let p_t denote the partial path between $e_{t'}(k)$ and $e_{t'}(k+1)$ in M, and p_c denote the partial path between $e_{c'}(k)$ and $e_{c'}(k+1)$ in M. Then, p_t and p_c are of the same length, say ℓ_p, and $M[p_t] \leq M[p_c]$: for their respective first elements, we have that $\mu_{t'}(k) \leq \mu_{c'}(k)$, and since they are horizontal or vertical paths, we know that for each $a \in [\ell_p]$, $M[p_t(a)] - M[p_c(a)] = \mu_{t'}(k) - \mu_{c'}(k)$. Hence, we can insert p_t between position k and $k+1$ in $e_{t'}$, and p_c between position k and $k+1$ in $e_{c'}$, and we do not violate the properties $M[e_{t'}] \leq M[e_{c'}]$ and $e_{t'} \ltimes e_{c'}$. $\square$

We prove the following strengthening of the Merge Dominator Lemma (Lemma 3.1).

Lemma 3.19 (Strong Merge Dominator Lemma). *Let r and c be integer sequence of length m and n, respectively. There exists a strong dominating non-diagonal merge of r and c, i.e. an integer sequence $t \in r \boxplus c$ such that $t \prec_{\ltimes} r \oplus c$, and this strong 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 $t \prec_{\ltimes} \tau(r) \oplus \tau(c)$.

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

Correctness of the above algorithm follows from Corollary 3.6 and Lemmas 3.16 and 3.18 which together guarantee that $\rho(t) \prec_{\ltimes} r \oplus c$, and by Observation 3.17, $\rho(t)$ is a non-diagonal merge, i.e. $\rho(t) \in a \boxplus b$. By Lemma 2.10, Step 1 can be done in time $\mathcal{O}(m+n)$, by Lemma 3.16, Step 2 takes time $\mathcal{O}(m+n)$ as well, and by Observation 3.17, 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 (Strong) Merge Dominator Lemma. All our algorithms, given a series parallel digraph G, 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. In the dynamic programming, we use the property that G_t is obtained either via series or parallel composition of the SPD's associated with its two children.

The remainder of the section is organized as follows. In Subsection 4.1, we provide an algorithm that computes the (weighted) cutwidth on (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. In Subsection 4.3, we give an $\mathcal{O}(n^2)$ time algorithm that computes the vertex separation number of a series parallel digraph. We would like to remark that all of the above algorithms make use of the fact that the dominating merge computed by the Strong Merge Dominator Lemma (Lemma 3.19) is non-diagonal, but only the algorithm computing the vertex separation number requires the strong domination property, for a technical reason that will become clear in Subsection 4.3.

4.1 Cutwidth

In this section we provide an $\mathcal{O}(n^2)$ time algorithm for the problem of computing the cutwidth of a series parallel digraph on n vertices. 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.

Cutwidth of Series Parallel Digraphs

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

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 π be a topological order of G. The sequence $x_1, \ldots, x_{n-1}$, where for $i \in [n-1]$,

$$x_i = |\{uv \in A(G) \mid \pi(u) \le i \wedge \pi(v) > i\}|,$$

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

To begin, we make a simple yet crucial observation that illustrates how the notion of domination of integer sequences is helpful in computing the cutwidth of directed acyclic graphs.

Observation 4.2. Let G be a graph and π, π' be linear orders of $V(G)$. If $\sigma(\pi) \prec \sigma(\pi')$, then $\mathsf{cutw}(\pi) \le \mathsf{cutw}(\pi')$.

The next lemma shows in combination with Observation 4.2 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. We will later use Lemma 3.19 to show that it is in fact enough to keep track of a *single* topological order at each stage of the algorithm.

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.

Lemma 4.3. *Let G_1 be an SPD and let π_1 and λ_1 be two topological orders of G_1 such that $\sigma(\pi_1) \prec \sigma(\lambda_1)$. Let G_2 be an SPD, and let $G = G_1 \vdash G_2$ or $G = G_1 \perp G_2$. Then, for each $\lambda \in \Pi(G)$ with $\lambda|_{V(G_1)} = \lambda_1$ there exists $\pi \in \Pi(G)$ with $\pi|_{V(G_1)} = \pi_1$ such that $\sigma(\pi) \prec \sigma(\lambda)$.*

Proof. If $G = G_1 \vdash G_2$, then we have that $\lambda = \lambda_1 \circ \lambda_2$ for some $\lambda_2 \in \Pi(G_2)$. Then, it is not difficult to see that $\pi = \pi_1 \circ \lambda_2$ is such that $\sigma(\pi) \prec \sigma(\lambda)$. Suppose $G = G_1 \perp G_2$, and let $n := |V(G)|$, and for $r \in [2]$, $n_r := |V(G_r)|$. We obtain $\pi \in \Pi(G)$ from π_1 by inserting the elements of $V(G_2)$ according to their position in λ. That is, π is obtained in such a way that for all i, whenever $\lambda(i) \in V(G_2)$, then $\lambda(i) = \pi(i)$. Now, let $\pi_2 := \pi|_{V(G_2)}$ and $\lambda_2 := \lambda|_{V(G_2)}$ and note that $\pi_2 = \lambda_2$ by construction. Let $\sigma(\pi) = x_1, \ldots, x_{n-1}$ and $\sigma(\lambda) = y_1, \ldots, y_{n-1}$. Let $i \in [n-1]$. Then, there is some $i_1 \in [n_1 - 1]$ and some $i_2 \in [n_2 - 1]$ such that x_i is the sum of $x^1_{i_1}$, the i_1-th element of $\sigma(\pi_1)$, and $x^2_{i_2}$, the i_2-th element of $\sigma(\pi_2)$, and y_i is the sum of $y^1_{i_1}$, the i_1-th element of $\sigma(\lambda_1)$ and $y^2_{i_2}$, the i_2-th element of $\sigma(\lambda_2)$. Since $\pi_2 = \lambda_2$, we have that $x^2_{i_2} = y^2_{i_2}$. Since $\sigma(\pi_1) \prec \sigma(\lambda_1)$, there are extensions e_1 of $\sigma(\pi_1)$ and f_1 of $\sigma(\lambda_1)$ of the same length, such that $e_1 \le f_1$. We can therefore conclude that there are extensions e of $\sigma(\pi)$ and f of $\sigma(\lambda)$ such that $e \le f$. ☐

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 $G_1 \vdash G_2$. Let $i^* := \pi^{-1}(t_2) = \pi^{-1}(s_1)$. Then, $\pi_1 := \pi(1), \ldots, \pi(i^*)$ is a topological order of G_1 and $\pi_2 := \pi(i^*), \pi(i^* + 1), \ldots, \pi(l(\pi))$ is a topological order of G_2. Moreover, $\sigma(\pi) = \sigma(\pi_1) \circ \sigma(\pi_2) \in \sigma(\Pi(G_1)) \odot \sigma(\Pi(G_2))$. The other inclusion can be shown by reading the previous argument backwards.

(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 $\pi_2 \in \Pi(G_2)$. Now, since

$\pi_1(1) = \pi_2(1)$ and $\pi_1(l(\pi_1)) = \pi_2(l(\pi_2))$, it immediately follows that the first element in $\sigma(\pi)$ is the sum of the first element of $\sigma(\pi_1)$ and the first element of $\sigma(\pi_2)$, and that the last element of $\sigma(\pi)$ is the sum of the last element of $\sigma(\pi_1)$ and the last element of $\sigma(\pi_2)$.

Generally, let $i \in \{2, \ldots, l(\pi) - 1\}$, and let i_1 be the maximum index such that $\pi(\pi_1^{-1}(i_1)) \leq i$, and define i_2 accordingly. Then, we have that the i-th element in $\sigma(\pi)$ is the sum of the i_1-th element in $\sigma(\pi_1)$ and the i_2-th element in $\sigma(\pi_2)$. Furthermore, the $(i+1)$-th element in π is either the $(i_1 + 1)$-th element in π_1, or the $(i_2 + 1)$-th element in π_2. This implies that the $(i+1)$-th element in $\sigma(\pi)$ is either the sum of the $(i_1 + 1)$-th element in $\sigma(\pi_1)$ and the i_2-th element in $\sigma(\pi_2)$, or the sum of the i_1-th element in $\sigma(\pi_1)$ and the $(i_2 + 1)$-th element in $\sigma(\pi_2)$. This concludes the proof that $\sigma(\pi) \in \sigma(\pi_1) \boxplus \sigma(\pi_2)$. The other inclusion can be shown similarly. $\square$

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 $|V(G)| =: n > 2$ and for the induction hypothesis that the claim holds for all SPD's on less than n vertices. 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 $G = G_1 \vdash G_2$. By the assumption that $\sigma(\pi_1^*)$ 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 2.7(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 $\sigma(\Pi(G))$ which will yield the lemma. 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 2.7(ii),[5] there exists some $f \in \sigma(\pi_1^*) \oplus \sigma(\pi_2^*)$ such that $f \prec e_1 + e_2$, and by Lemma 3.7, 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:

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

[5]Take $a = e_1$, $b = e_2$, $a_0 = \sigma(\pi_1)$, and $b_0 = \sigma(\pi_2)$.

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. [23] 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 children ℓ and 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 children ℓ and 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 Strong Merge Dominator Lemma (Lemma 3.19). 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. Observation 4.2 and Lemma 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 3.19, we have that the total runtime of the algorithm is $\mathcal{O}(n^2)$. □

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 $\mathsf{wcutw}(G, \omega) := \min_{\pi \in \Pi(G)} \mathsf{wcutw}(\pi, \omega)$.

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.

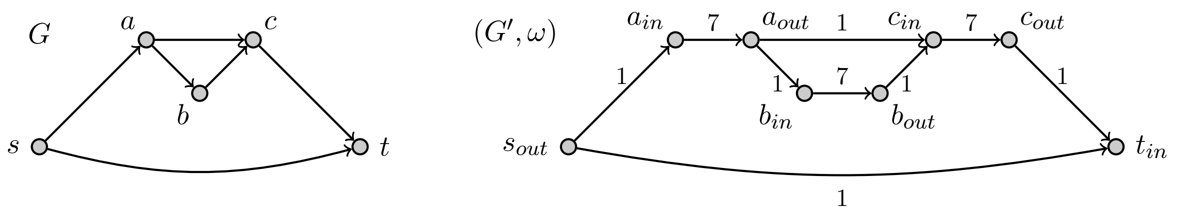


Figure 11: Illustration of the transformation given in the proof of Theorem 4.9. Note that in this case, $m = 6$, so the arcs between vertices v_{in} and v_{out} have weight 7.

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 reduction to the Weighted Cutwidth of SPD's problem. We would like to remark that this reduction is similar to one provided in [6], however some modifications are necessary to ensure that the digraph resulting from the reduction 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 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 $|A(G')| \leq |A(G)| + |V(G)|$.

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 $n > 2$ and for the induction hypothesis that the claim holds for all $n' < n$. 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. $\diamond$

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 $\mathsf{mcutw}(\pi) \leq k$. 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 $\mathsf{mcutw}(\pi) \leq k$, 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 $\mathsf{mcutw}(G) \leq k$, 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 $2k + 1 \leq m + k + 1$. $\diamond$

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 $\mathsf{wcutw}(\pi', \omega) \le m+k+1$, we can conclude that in π, there are at most $(m+k+1) - (m-1) = k$ arcs crossing the vertex v in π. $\diamond$

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 $\mathcal{O}(n^2)$. $\square$

4.3 Vertex Separation Number

In this section we provide an $\mathcal{O}(n^2)$ time algorithm for the problem of computing the vertex separation number of a series-parallel digraph on n vertices. Recall that given a topological order $v_1, \ldots, v_n$ of a directed acyclic graph G, its vertex separation number is the maximum over all $i \in [n-1]$, of the number of vertices in $\{v_1, \ldots, v_i\}$ that have a neighbor in $\{v_{i+1}, \ldots, v_n\}$. The vertex separation number of a directed acyclic graph is the minimum vertex separation number over all its topological orders.

Vertex Separation of Series-Parallel Digraphs

Input:	A series-parallel digraph G.
Question:	What is the vertex separation number of G?

To make this problem amenable to be solved using merges of integer sequences, we define a notion of a VSN sequence of a topological order of a series-parallel digraph. Note that in contrast to cutwidth and modified cutwidth that count arcs crossing a cut, we are now counting the smaller-numbered vertices having arcs crossing a cut. Since in the parallel composition $(G, (s,t)) = (G_1, (s_1, t_1)) \perp (G_2, (s_2, t_2))$ we identify the sources of G_1 and G_2 by $s = s_1 = s_2$ the merge of a sequence from G_1 and one from G_2 must handle the source vertex in a special way. To this end we define the VSN sequence so that each vertex apart from the source vertex contributes a value of two while the source vertex contributes only a value of one, if at all.

Definition 4.10 (VSN Sequence). Let $(G, (s,t))$ be a series-parallel digraph on n vertices and let $\omega \colon V(G) \to \{1,2\}$ be a weight function assigning $\omega(s) = 1$ and $\omega(v) = 2$ for all $v \in V(G) \setminus \{s\}$. Let π be a topological order of G. The sequence $x_1, \ldots, x_{n-1}$, where for $i \in [n-1]$,

$$x_i = \sum_{v \in C_i} \omega(v), \text{ where } C_i = \{u \in V(G) \mid \exists v \in V(G) \colon uv \in A(G) \wedge \pi(u) \le i \wedge \pi(v) > i\}$$

is the *VSN sequence* of π, and denoted by $\alpha(\pi)$. For a set of topological orders $\Pi' \subseteq \Pi(G)$, we let $\alpha(\Pi') := \{\alpha(\pi) \mid \pi \in \Pi'\}$, and we use the shorthand '$\alpha(G)$' for '$\alpha(\Pi(G))$'.

Since we can assume that the source vertex has at least one arc, the VSN sequence will have a prefix consisting of odd numbers, ending at the largest index of any neighbor of s in the topological order, followed by a suffix consisting of even numbers. Note the even suffix will be empty in case $st \in A(G)$. To handle the odd prefix of VSN sequences correctly in the following algorithm, we define the following notions.

Definition 4.11. Let s be an integer sequence of length n. The *odd (even) prefix* of s is the longest prefix of s consisting only of odd (even) numbers.

Let m denote the length of the odd prefix of s (possibly $m = 0$). Then, $\phi(s) := s'(1), \ldots, s'(m)$, where for all $i \in [n]$, $s'(i) = s(i) + 1$ if $i \leq m$, and $s'(i) = s(i)$, otherwise. Now let m denote the length of the even prefix of s (possibly $m = 0$). Then, $\psi(s) := s'(1), \ldots, s'(m)$, where for all $i \in [n]$, $s'(i) = s(i) - 1$ if $i \leq m$ and $s'(i) = s(i)$, otherwise. For a set S of integer sequences, we let $\phi(S) := \{\phi(s) \mid s \in S\}$ and $\psi(S) := \{\psi(s) \mid s \in S\}$.

We can now use the previously defined functions to describe the VSN sequences of an SPD G in terms of the smaller SPD's it is obtained from. Intuitively, the following lemma says that if G is obtained by series composition of G_1 and G_2, then the set of VSN sequences is precisely the set of all sequences that are obtained via concatenation of a VSN sequence of G_1 and a VSN sequence of G_2, to whose odd prefix we added a value of 1; and if G is obtained by parallel composition of G_1 and G_2, then the set of VSN sequences of G is obtained from the set of all non-diagonal merges of the VSN sequences of G_1 and G_2, and subtracting a value of 1 from the even prefix of each of its members.

Lemma 4.12. *Let G_1 and G_2 be SPD's. Then the following hold.*

(i) $\alpha(G_1 \vdash G_2) = \alpha(G_1) \odot \phi(\alpha(G_2))$.

(ii) $\alpha(G_1 \perp G_2) = \psi(\alpha(G_1) \boxplus \alpha(G_2))$.

Proof. The proof can be done in the same way as the one for Lemma 4.4, with the following differences. For (i), we observe that s_2 (being the source of G_2) only accounts for a value of 1 in each VSN sequence $\alpha(\pi_2)$ in $\alpha(G_2)$. In G however, s_2 is not a source vertex anymore, so in each cut affected by s_2 it contributes with a value of 2 to it. The cuts affected by s_2 are precisely the ones that correspond to the odd prefix in $\alpha(\pi_2)$, hence taking $\phi(\alpha(\pi_2))$ instead yields the correct VSN sequence for G.

For (ii), the only difference with the argument given in the proof of Lemma 4.4(ii) is that the contribution of the source s of G is not counted correctly when considering the non-diagonal merges of $\alpha(G_1)$ with $\alpha(G_2)$. To see this, let $\alpha(\pi_1) \in \alpha(G_1)$ and $\alpha(\pi_2) \in \alpha(G_2)$, and consider the merge matrix M of $\alpha(\pi_1)$ and $\alpha(\pi_2)$. (See Figure 12 for an illustration.) Let i_1^* denote the largest index (in π_1) of any neighbor of s_1 in G_1, and let i_2^* denote the largest index (in π_2) of any neighbor of s_2 in G_2. In the submatrix $M^* := M[1..i_1^*, 1..i_2^*]$, s_1 contributes a value of 1 to each entry, and s_2 contributes a value of 1 to each entry. Each non-diagonal path in M corresponds to a topological order in G. However, by the observation just made, the values in M^* overcount the contribution of $s = s_1 = s_2$ by a value of 1 each.

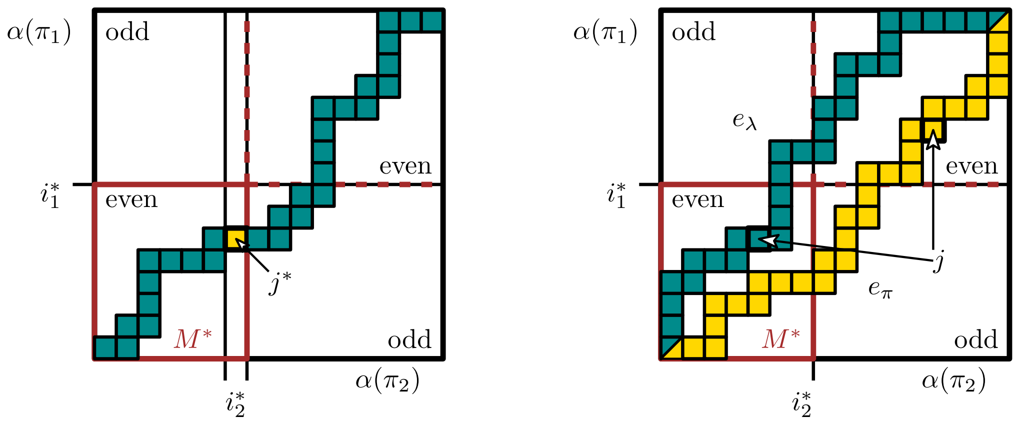


Figure 12: On the left side, an illustration of the proof of Lemma 4.12(ii). Note that on the depicted path, all elements up to position j^* are even and the element at $j^* + 1$ is odd. On the right side, an illustration of the final contradiction obtained in the proof of Lemma 4.13(ii). Note that only the elements on the paths corresponding to the extensions are shown, and repetitions of elements are omitted.

Let p be any path in M, and let π be the corresponding topological order of G. We will argue that $\psi(M[p]) = \alpha(\pi)$. Let j^* denote the last index (in p) such that $p(j^*) \in [i_1^*] \times [i_2^*]$, and let $p^* := p[1..j^*]$. We observe two things: first, the values in $M[p^*]$ are precisely the ones being miscounted, namely overcounted by 1, and as $M^* := M[1..i_1^*, 1..i_2^*]$ contains the sums of elements of the odd prefix of $\alpha(\pi_1)$ and the odd prefix of $\alpha(\pi_2)$, all of its entries are even. We argue that $M[p(j^* + 1)]$ must be odd. All entries in $M[1..i_1^*, (i_2^* + 1)..n]$ are odd, as they are the sum of an element of the odd prefix of $\alpha(\pi_1)$ and an element of the even suffix of $\alpha(\pi_2)$, and symmetrically, all entries in $M[(i_1^* + 1)..m, 1..i_2^*]$ are odd. So, for $M[p(j^* + 1)]$ to be even, $p(j^* + 1)$ must be in $[(i_1^* + 1)..m] \times [(i_2^* + 1)..n]$, so j^* must be equal to (i_1^*, i_2^*). However, then p contains a diagonal step. We have argued that $M[p^*]$ is the even prefix of $M[p]$, so indeed $\psi(M[p]) = \alpha(\pi)$. □

Lemma 4.13. *Let G be an SPD without parallel arcs. Then there is a topological order π^* of G such that $\alpha(\pi^*)$ strongly dominates all VSN 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 $\alpha(\pi_r^*)$ strongly dominates all VSN 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 $\alpha(\pi^*)$ strongly dominates $\alpha(\pi_1^*) \boxplus \alpha(\pi_2^*)$.

Proof. We prove the lemma by induction on $n := |V(G)|$. For $n = 2$ it is trivial. Suppose for the induction hypothesis that the lemma holds for all $n' < n$ and suppose that $n > 2$. For $r \in [2]$, let $n_r := |V(G_r)|$ and note that since G has no parallel arcs, $n_r < n$. Hence, by the induction hypothesis there is a topological order $\pi_r^* \in \Pi(G_r)$ such that $\alpha(\pi_r^*) \prec_{\ltimes} \alpha(G_r)$.

For (i), we have to show that under its assumptions, $\phi(\alpha(\pi_2^*)) \prec_{\ltimes} \phi(\alpha(G_2))$, then the result follows from Lemma 2.7(v) (which also holds for strong domination), and Lemma 4.12(ii). Let $\alpha(\lambda_2) \in \alpha(G_2)$. For convenience, let $\alpha_\pi := \alpha(\pi_2^*)$ and $\alpha_\lambda := \alpha(\lambda_2)$. There are extensions e_π of

α_π and e_λ of α_λ of the same length ℓ such that $e_\pi \leq e_\lambda$ and $e_\pi \ltimes e_\lambda$. Let e'_π and e'_λ be obtained from e_π and e_λ by adding 1 to each element of the corresponding odd prefix. We claim that e'_π and e'_λ witness that $\phi(\alpha_\pi) \prec_\ltimes \phi(\alpha_\lambda)$. It is clear that the strong domination property remains intact. Suppose there is some $i \in [\ell]$ such that $e'_\pi(i) > e'_\lambda(i)$. Since $e_\pi(i) \leq e_\lambda(i)$, this means that $e'_\pi(i) = e_\pi(i) + 1$, and $e'_\lambda(i) = e_\lambda(i)$. However, this implies that $e_\pi(i)$ was odd, and $e_\lambda(i)$ was even, and so $e_\pi(i) < e_\lambda(i)$, implying $e'_\pi(i) \leq e'_\lambda(i)$.

For (ii), we have to show that under its assumptions, $\alpha(\pi^*) \prec_\ltimes \alpha(G)$. Let $\alpha(\lambda) \in \alpha(G)$. By Lemma 4.12(ii), we know that there are $\alpha(\lambda_1) \in \alpha(G_1)$ and $\alpha(\lambda_2) \in \alpha(G_2)$ such that $\alpha(\lambda) \in \psi(\alpha(\lambda_1) \boxplus \alpha(\lambda_2))$. By the argument given in the proof of Lemma 4.5(ii),[6] we know that there are extensions e_1 of $\alpha(\lambda_1)$ and e_2 of $\alpha(\lambda_2)$ of the same length ℓ such that $\alpha(\pi^*) \prec_\ltimes e_1 + e_2$. Hence, there is an extension e_π of $\alpha(\pi^*)$ and e_λ of $e_1 + e_2$ of the same length ℓ^* such that $e_\pi \leq e_\lambda$ and $e_\pi \ltimes e_\lambda$. We argue that e_π and e_λ also witness that $\alpha(\pi^*) \prec_\ltimes \psi(e_1+e_2)$, which will finish the proof.

Let $e'_\pi := \psi(e_\pi)$ and $e'_\lambda := \psi(e_\lambda)$. Again it is clear that $e'_\pi \ltimes e'_\lambda$. Suppose there is some $j \in [\ell^*]$ such that $e'_\pi(j) > e'_\lambda(j)$. Since $e_\pi(j) \leq e_\lambda(j)$, this must mean that $e'_\pi(j) = e_\pi(j)$ and $e'_\lambda(j) = e'_\lambda(j) - 1$. Let $M^* := M[1..i_1^*, 1..i_2^*]$ be as in the proof of Lemma 4.12(ii), and see Figure 12 for an illustration. Under our stated conditions, we have that $e_\pi(j)$ is an element of $M[(i_1^* + 1)..m, (i_2^* + 1)..n]$, and $e_\lambda(j)$ is an element of M^* (since both values are even, and the function ψ subtracted one only from $e_\lambda(j)$ and not from $e_\pi(j)$). However, this contradicts the fact that $e_\pi \ltimes e_\lambda$. □

Theorem 4.14. *Let $(G, (s,t))$ be an SPD on n vertices. There is an algorithm that computes in time $\mathcal{O}(n^2)$ the vertex separation number of G, and outputs a topological ordering that achieves the upper bound.*

Proof. We may assume that G has no parallel arcs; for if there are several parallel arcs between a pair of vertices, we can remove all but one of them without changing the vertex separation number. FWe use the algorithm of Valdes et al. [23] 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 such that $\alpha(\pi_t)$ strongly dominates all VSN 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 children ℓ and r). In this case, we look up π_ℓ, a topological order such that $\alpha(\pi_\ell)$ strongly dominates all VSN sequences of G_ℓ, and π_r, a topological order such that $\alpha(\pi_r)$ strongly dominates all VSN sequences of G_r. Following Lemma 4.13(i), we return $\pi_\ell \circ \pi_r$.

Case 3 (t is a parallel node with children ℓ and r). In this case, we look up π_ℓ and π_r as in Case 2, and compute a strong dominating merge $\alpha(\pi_t)$ of $\alpha(\pi_\ell)$ and $\alpha(\pi_r)$ using the Strong Merge Dominator Lemma (Lemma 3.19). Following Lemma 4.13(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. Correctness of Case 1 is immediate, and correctness of Case 2 and 3 follow from Lemma 4.13(i) and Lemma 4.13(ii), respectively, together with the analogue of Lemma 4.3 for VSN sequences. Since T has $\mathcal{O}(n)$ nodes

[6]Note that Lemma 2.7(ii) holds for strong domination as well.

and each of the above cases can be handled in at most $\mathcal{O}(n)$ time by Lemma 3.19, we have that the total runtime of the algorithm is $\mathcal{O}(n^2)$. ☐

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, e.g., for which width measures can we give XP algorithms when parameterized by the treewidth of the input graph?

References

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

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

[3] Hans L. Bodlaender. A partial k-arboretum for graphs of bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998.

[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] Christoph W. Kessler. Scheduling expression DAGs for minimal register need. *Computer Languages*, 24(1):33–53, 1998.

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

[16] 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.

[17] 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.

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

[19] Ravi Sethi. Complete register allocation problems. *SIAM Journal on Computing*, 4(3):226–248, 1975.

[20] Ravi Sethi and Jeffrey D. Ullman. The generation of optimal code for arithmetic expressions. *Journal of the ACM*, 17(4):715–728, 1970.

[21] 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.

[22] 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.

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