# Typical Sequences Revisited – Computing Width Parameters of Graphs

**Hans L. Bodlaender**
Department of Computer Science, Utrecht University, The Netherlands
h.l.bodlaender@uu.nl

**Lars Jaffke**
Department of Informatics, University of Bergen, Norway
lars.jaffke@uib.no

**Jan Arne Telle**
Department of Informatics, University of Bergen, Norway
jan.arne.telle@uib.no

**Abstract**

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

**2012 ACM Subject Classification** Mathematics of computing → Combinatorial algorithms; Mathematics of computing → Graph algorithms

**Keywords and phrases** typical sequences, treewidth, series parallel digraphs, cutwidth, modified cutwidth

**Digital Object Identifier** 10.4230/LIPIcs.STACS.2020.57

**Related Version** A full version of the paper is available at `https://arxiv.org/abs/1905.03643`.

**Funding** *Hans L. Bodlaender*: Partially supported by the Networks project, funded by the Netherlands Organization for Scientific Research (NWO).
*Lars Jaffke*: Supported by the Bergen Research Foundation (BFS).

**Acknowledgements** 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.

## 1 Introduction

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

The notion of typical sequences was introduced independently in 1991 by Lagergren and Arnborg [15] and Bodlaender and Kloks [8]. In both papers, it is a key element in an explicit dynamic programming algorithm that given a tree decomposition of bounded width $\ell$, 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 [3] for treewidth-$k$. If one analyses the running time of Bodlaender's algorithm for treewidth or pathwidth $\leq k$, then one can observe that the bottleneck is in the subroutine that calls the Bodlaender-Kloks dynamic programming subroutine, with both the subroutine and the main algorithm having time $\mathcal{O}(2^{\mathcal{O}(k^3)}n)$ for treewidth, and $\mathcal{O}(2^{\mathcal{O}(k^2)}n)$ for pathwidth. See also the recent work by Fürer for pathwidth [13], and the simplified versions of the algorithms of [3, 8] by Althaus and Ziegler [1]. Now, over a quarter of a century after the discovery of these results, even though much work has been done on treewidth recognition algorithms (see e.g. [2, 5, 11, 12, 13, 14, 16, 17]), these bounds on the function of $k$ are still the best known, i.e. no $\mathcal{O}(2^{o(k^3)}n^{O(1)})$ algorithm for treewidth, and no $\mathcal{O}(2^{o(k^2)}n^{O(1)})$ algorithm for pathwidth is known. An interesting question, and a long-standing open problem in the field [4, Problem 2.7.1], is whether such algorithms can be obtained. Possible approaches to answer such a question is to design (e.g. ETH or SETH based) lower bounds, find an entirely new approach to compute treewidth or pathwidth in a parameterized setting, or improve upon the dynamic programming algorithms of [15] and [8]. Using our Merge Dominator Lemma we can go one step towards the latter, as follows.

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

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

We give direct algorithmic consequences of the Merge Dominator Lemma in the realm of computing width parameters of directed acyclic graphs (DAGs). Specifically, we show that the (Weighted) Cutwidth and Modified Cutwidth problems on DAGs, which given a directed acyclic graph on $n$ vertices, ask for the topological order that minimizes the

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

*cutwidth* and *modified cutwidth*, respectively, can be solved in $\mathcal{O}(n^2)$ time on *series parallel digraphs*. Note that the restriction of the solution to be a *topological* order has been made as well in other works, e.g. [6].

Our algorithm for Cutwidth of series parallel digraphs has the same structure as the dynamic programming algorithm for undirected Cutwidth [6], but, in addition to obeying directions of edges, we have a step that only keeps characteristics that are not dominated by another characteristic in a table of characteristics. Now, with help of our Merge Dominator Lemma, we can show that in the case of series parallel digraphs, there is a unique dominating characteristic; the dynamic programming algorithm reverts to computing for each intermediate graph a single "optimal partial solution". This strategy also works in the presence of edge weights, which gives the algorithm for the corresponding Weighted Cutwidth problem on series parallel digraphs. Note that the cutwidth of a directed acyclic graph is at least the maximum indegree or outdegree of a vertex; e.g., a series parallel digraph formed by the parallel composition of $n-2$ paths with three vertices has $n$ vertices and cutwidth $n-2$. To compute the *modified* cutwidth of a series parallel digraph, we give a linear-time reduction to the Weighted Cutwidth problem on series parallel digraphs.

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

Statements marked with "♣" are proved in the full version of the paper.

## 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* $\pi$ *to* $X'$ as the bijection $\pi|_{X'} \colon X' \to [n']$ which is such that for all $x', y' \in X'$, $\pi|_{X'}(x') < \pi|_{X'}(y')$ if and only if $\pi(x') < \pi(y')$.

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

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

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

be the corresponding sequence of entries from $M$.

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

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

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

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

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

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

▶ **Definition 1.** *Let $r$ and $s$ be two integer sequences of the same length $n$.*
**(i)** *If for all $i \in [n]$, $r(i) \le s(i)$, then we write "$r \le s$".*
**(ii)** *We write $q = r + s$ for the integer sequence $q(1), \dots, q(n)$ with $q(i) = r(i) + s(i)$ for all $i \in [n]$.*

▶ **Definition 2** (Extensions). *Let $s$ be a sequence of length $n$. We define the set $E(s)$ of* extensions *of $s$ as the set of sequences that are obtained from $s$ by repeating each of its elements an arbitrary number of times, and at least once. Formally, we let*

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

▶ **Definition 3** (Domination). *Let $r$ and $s$ be integer sequences. We say that $r$* dominates *$s$, in symbols "$r \prec s$", if there are extensions $r^* \in E(r)$ and $s^* \in E(s)$ of the same length such that $r^* \le s^*$. If $r \prec s$ and $s \prec r$, then we say that $r$ and $s$ are* equivalent*, and we write $r \equiv s$.*

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

▶ Remark 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 5** (Merges). *Let $r$ and $s$ be two integer sequences. We define the set of all* merges *of $r$ and $s$, denoted by $r \oplus s$, as $r \oplus s := \{r^* + s^* \mid r^* \in E(r), s^* \in E(s), l(r^*) = l(s^*)\}$.*

## 2.1 Typical Sequences

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

▶ **Definition 6.** *Let $s = s(1), \dots, s(n)$ be an integer sequence of length $n$. The* typical sequence *of $s$, denoted by $\tau(s)$, is obtained from $s$ by an exhaustive application of the following two operations:*

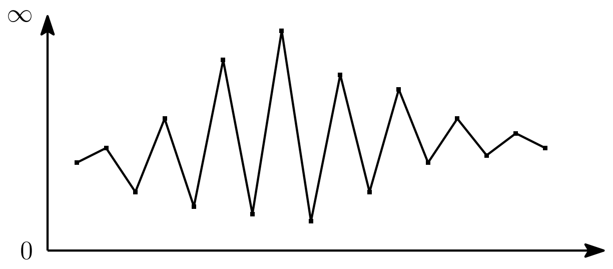

**Figure 1** Illustration of the shape of a typical sequence.

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

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

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

▶ **Proposition 7** (♣). *Let $s$ be an integer sequence and let $i^\star \in \{\text{argmin}^\star(s), \text{argmax}^\star(s)\}$. Let $1 =: j_0 < j_1 < j_2 < \ldots < j_t < j_{t+1} := i^\star$ be pairwise distinct integers, such that $s(j_0), \ldots, s(j_{t+1})$ are pairwise distinct. If for all $h \in [0..t]$,*
- *if $s(j_h) > s(j_{h+1})$ then $j_h = \text{argmax}^\star(s[1..j_{h+1}])$ and $j_{h+1} = \text{argmin}^\star(s[1..j_{h+1}])$, and*
- *if $s(j_h) < s(j_{h+1})$ then $j_h = \text{argmin}^\star(s[1..j_{h+1}])$ and $j_{h+1} = \text{argmax}^\star(s[1..j_{h+1}])$,*

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

The idea of the algorithm is as follows. First, it is immediate that the typical sequence of $s$ must contain its minimum and its maximum. We then observe the structure of $\tau(s)$ between $i^\star := \min \text{argmin}(s) \cup \text{argmax}(s)$ and $k^\star := \max \text{argmin}(s) \cup \text{argmax}(s)$. Next, we find a set of indices from $[i^\star]$ that satisfy the preconditions of Proposition 7 which gives indices inducing the typical sequence on $s[1..i^\star]$. By symmetry, we can again use Proposition 7 to find the indices inducing $\tau(s)$ on $s[k^\star..n]$.

▶ **Lemma 8** (♣). *Let $s$ be an integer sequence of length $n$. Then, one can compute $\tau(s)$, the typical sequence of $s$, in time $\mathcal{O}(n)$.*

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

▶ **Lemma 9** (Bodlaender and Kloks [8]). *Let $r$ and $s$ be two integer sequences.*

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

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

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

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

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

## 2.2 Directed Acyclic Graphs

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

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

▶ **Definition 10.** *Let $G$ be a directed acyclic graph and let $\pi \in \Pi(G)$.*

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

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

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

We now introduce series parallel digraphs. Note that the following definition coincides with the notion of "edge series-parallel multidigraphs" in [20].

▶ **Definition 11** (Series Parallel Digraph (SPD))**.** *A (multi-)digraph $G$ with an ordered pair of* terminals $(s, t) \in V(G) \times V(G)$ *is called* series parallel digraph (SPD)*, often denoted by $(G, (s, t))$, if one of the following hold.*

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

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

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

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

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

▶ **Theorem 12** (Valdes et al. [20]). *Let $G$ be a directed graph on $n$ vertices and $m$ arcs. There is an algorithm that decides in time $\mathcal{O}(n+m)$ whether $G$ is a series parallel digraph and if so, it outputs a decomposition tree that yields $G$.*

## 3 The Merge Dominator Lemma

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

▶ **Lemma 13** (Merge Dominator Lemma). *Let $r$ and $c$ be integer sequences 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.** 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 19 in Section 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 Section 3.3 where we prove the "Chop Lemmas" that come in two versions. The "bottom version" (Lemma 20) shows that in $M_1$, we can find a dominating path by repeatedly chopping away the *last* two rows or columns and remembering a vertical or horizontal length-2 path. The "top version" (Corollary 21) is the symmetric counterpart for $M_2$. The proofs of the Chop Lemmas only hold when $r$ and $c$ are *typical sequences*, and in Section 3.4 we present the "Split-and-Chop Algorithm" that computes a dominating path in a merge matrix of two typical sequences. Finally, in Section 3.5, we generalize this result to arbitrary integer sequences, using the Split-and-Chop Algorithm and one additional construction. ◀

### 3.1 The Merge Matrix, Paths, and Non-Diagonality

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

▶ **Definition 14** (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 15** (Path in a Matrix). *Let $M$ be an $m \times n$ matrix. A* path *in $M$ is a sequence $p(1), \ldots, p(\ell)$ of indices from $M$ such that*

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

**(ii)** *for $t \in [\ell - 1]$, 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 $M$. For two paths $p, q \in \mathcal{P}(M)$, we may simply say that $p$* dominates *$q$, if $M[p]$ dominates $M[q]$.*

*A path $p(1), \ldots, p(\ell)$ is called* non-diagonal *if the second condition is replaced by the following.*

**(ii)'** *For $t \in [\ell - 1]$, let $p(t) = (i,j)$; then, $p(t+1) \in \{(i+1,j), (i,j+1)\}$.*

In analogy with extensions of integer sequences, an extension $e$ of a path $p$ in a matrix $M$ is as well a sequence of indices of $M$, and we again denote the corresponding integer sequence by $M[e]$. A consequence of Lemma 9(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 16.** *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 Section 3.5, and in the algorithmic applications of the Merge Dominator Lemma given in Section 4. For two integer sequences $r$ and $s$, we denote by $r \boxplus s$ the set of all *non-diagonal merges* of $r$ and $s$, which are not allowed to have "diagonal" steps: we have that for all $t \in r \boxplus s$ and all $i \in [l(t) - 1]$, if $t(i) = r(i_r) + s(i_s)$, then $t(i+1) \in \{r(i_r + 1) + s(i_s), r(i_r) + s(i_s + 1)\}$. We now show that for each merge that uses diagonal steps, there is always a non-diagonal merge that dominates it.

▶ **Lemma 17.** *Let $r$ and $s$ be two integer sequences of length $m$ and $n$, respectively. For any merge $q \in r \oplus s$, there is a non-diagonal merge $q' \in r \boxplus s$ such that $q' \prec q$. Furthermore, given $q$, $q'$ can be found in time $\mathcal{O}(m + n)$.*

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. These paths appear in the following special cases of the Merge Dominator Lemma, which will be crucial for the proof of the Split Lemma.

▶ **Lemma 18.** *Let $r$ and $c$ be integer sequences of length $m$ and $n$, respectively, and let $M$ be the merge matrix of $r$ and $c$. Let $i \in \operatorname{argmin}(r)$ and $j \in \operatorname{argmin}(c)$.*
**(i)** *If $i = 1$ and $j = n$, then $p_{\lrcorner}$ dominates all paths in $M$, i.e. $p_{\lrcorner} \prec \mathcal{P}(M)$.*
**(ii)** *If $i = m$ and $j = 1$, then $p_{\ulcorner}$ dominates all paths in $M$, i.e. $p_{\ulcorner} \prec \mathcal{P}(M)$.*

**Proof.** (i) For an illustration of this proof see the left side of 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 $p_{\lrcorner}$ dominates $q$.

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

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

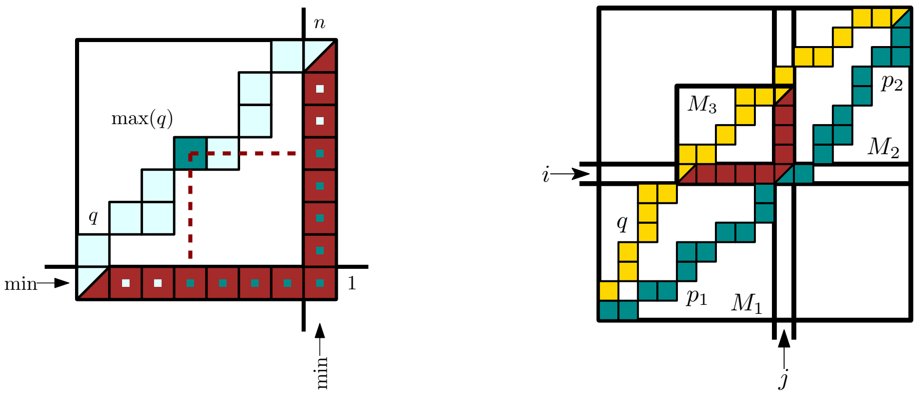

**Figure 2** Illustration of the proof strategy of the Split Lemma (Lem. 19).

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

## 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 dominating path for each of them, and paste them together to obtain a dominating path for $M$.

▶ **Lemma 19** (Split Lemma). *Let $r$ and $c$ be integer sequences of length $m$ and $n$, respectively, and let $M$ be the merge matrix of $r$ and $c$. Let $i \in \text{argmin}(r)$ and $j \in \text{argmin}(c)$. Let $M_1 := M[1..i, 1..j]$ and $M_2 := M[i..m, j..n]$ and for all $h \in [2]$, let $p_h \in \mathcal{P}(M_h)$ be a dominating path in $M_h$, i.e. $p_h \prec \mathcal{P}(M_h)$. Then, $p_1 \circ p_2$ is a dominating path in $M$, i.e. $p_1 \circ p_2 \prec \mathcal{P}(M)$.*

**Proof.** Let $q$ be any path in $M$. If $q$ contains $(i, j)$, then $q$ has two consecutive parts, say $q_1$ and $q_2$, such that $q_1 \in \mathcal{P}(M_1)$ and $q_2 \in \mathcal{P}(M_2)$. Hence, $p_1 \prec q_1$ and $p_2 \prec q_2$, so by Lemma 9(v), $p_1 \circ p_2 \prec q_1 \circ q_2$.

Now let $p := p_1 \circ p_2$ and suppose $q$ does not contain $(i, j)$. Then, $q$ either contains some $(i, j')$ with $j' < j$, or some $(i', j)$, for some $i' < i$. We show how to construct extensions of $p$ and $q$ that witness that $p$ dominates $q$ in the first case, and remark that the second case can be shown symmetrically. We illustrate this situation in the right side of Figure 2.

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

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

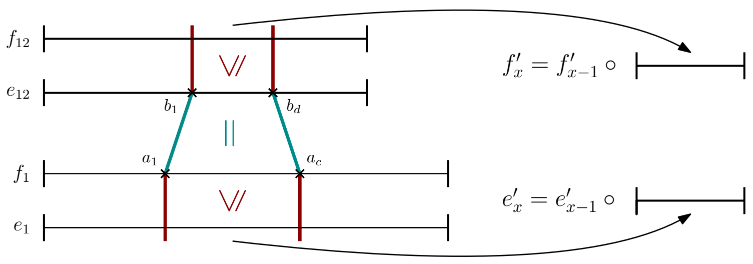

**Figure 3** Constructing extensions in the proof of Lemma 19.

Since $q_1^+ \in \mathcal{P}(M_1)$ and $p_1 \prec \mathcal{P}(M_1)$, we have that $p_1 \prec q_1^+$, similarly that $p_2 \prec q_2^+$ and considering $M_3 := M[i'..i, j..j']$, we have by Lemma 18(i) that $p_{12} := p_\lrcorner(M_3) = (i,j'),(i,j'+1),\ldots,(i,j),(i+1,j),\ldots,(i',j)$ dominates $q_{12}$. Consequently, we consider the following extensions of these sequences.

**(I)** We let $e_1 \in E(p_1)$ and $f_1 \in E(q_1^+)$ such that $l(e_1) = l(f_1)$ and $M[e_1] \leq M[f_1]$.
**(II)** We let $e_{12} \in E(p_{12})$, and $f_{12} \in E(q_{12})$ such that $l(e_{12}) = l(f_{12})$ and $M[e_{12}] \leq M[f_{12}]$.
**(III)** We let $e_2 \in E(p_2)$, and $f_2 \in E(q_2^+)$ such that $l(e_2) = l(f_2)$ and $M[e_2] \leq M[f_2]$.

We construct extensions $e' \in E(p)$ and $f' \in E(q)$ as follows. Let $z$ be the last index in $q$ of any element that is matched up with $(i,j)$ in the extensions of (II). (Following the proof of Lemma 18, this would mean $z$ is the index of $\max(q_{12})$ in $q$.) We first construct a pair of extensions $e'_j \in E(p_1)$, and $f'_j \in E(q[1..z])$ with $l(e'_j) = l(f'_j)$ and $M[e'_j] \leq M[f'_j]$. With a symmetric procedure, we can obtain extensions of $p_2$ and of $q[(z+1)..l(q)]$, and use them to obtain extensions of $p = p_1 \circ p_2$ and $q = q[1..z] \circ q[(z+1)..l(q)]$ witnessing that $p \prec q$.

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

For $x = j', j'+1, \ldots, j$, we inductively construct $e'_x$ and $f'_x$ using $e'_{x-1}$ and $f'_{x-1}$, for an illustration see Figure 3. We maintain as an invariant that $l(e'_{x-1}) = l(f'_{x-1})$ and that $M[e'_{x-1}] \leq M[f'_{x-1}]$. Let $a_1, \ldots, a_c$ denote the indices of the occurrences of $(i,x)$ in $f_1$, and $b_1, \ldots, b_d$ denote the indices of the occurrences of $(i,x)$ in $e_{12}$. We let:

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

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

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

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

## 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

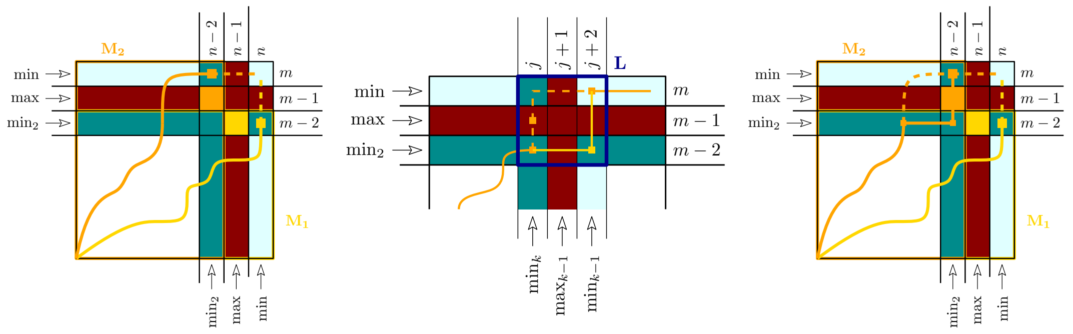

**Figure 4** Illustration of the ideas in the main step of the proof of the bottom version of the Chop Lemmas (Lem. 20).

Lemmas”. It will turn out that when applied to typical sequences, a repeated application of these lemmas yields a dominating path in $M$. This insight crucially helps in arguing that the dominating path in a merge matrix can be found in *linear* time. 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 Section 3.5 we will generalize the results that rely on the following lemmas to arbitrary integer sequences.

▶ **Lemma 20** (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 $m \in \text{argmin}(r)$ and $n \in \text{argmin}(c)$ and let $M_1 := M[1..(m-2), 1..n]$ and $M_2 := M[1..m, 1..(n-2)]$ and for all $h \in [2]$, let $p_h \prec \mathcal{P}(M_h)$. Let $p_1^+ := p_1 \circ (m-1, n), (m, n)$ and $p_2^+ := p_2 \circ (m, n-1), (m, n)$.*

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

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

**Outline of the proof.** We first argue that each path in $M$ is dominated by at least one of $p_1^+$ and $p_2^+$. If a path contains either $(m-2, n)$, or $(m, n-2)$, then it is easily seen that it is dominated by $p_1^+$ or $p_2^+$, respectively. If a path does not contain either of them, then there always is a path that dominates it, and contains either $(m-2, n)$ or $(m, n-2)$. This is due to the fact that $m-1$ is a maximum row and $n-1$ a maximum column.

Next, consider the setting of (i), and for an illustration see Figure 4, beginning with the left hand side. Most effort is spent in proving that $p_1^+$ dominates $p_2^+$ under the stated assumption. Towards this claim, we first show that we can find a path $p_2'$ in $M_2$ that uses $(m-2, n-2)$, and dominates $p_2$. This is done by considering situations such as in the middle of Figure 4. Assume the notation used there, and note that $p_2$ uses the element $(m, j)$. Since $M[m-2, n-1] \leq M[m-1, n-2]$, and using the structure of typical sequences ending in the minimum element, we can conclude that $M[m-2, j+1] \leq M[m-1, j]$. We then show that in the $3 \times 3$-submatrix $L$, the corner path $p_\lrcorner(L)$ is a dominating path. Using the corresponding extensions, we show that the path obtained from $p_2$ by replacing the $p_\ulcorner(L)$ path with the $p_\lrcorner(L)$ path, dominates $p_2$. Iteratively applying such arguments and transitivity of “$\prec$” lets us conclude that there is a path $p_2'$ in $M_2$ that uses $(m-2, n-2)$, and dominates $p_2$, see the right hand side of Figure 4. Now, let $p_2''$ be the subpath of $p_2'$ ending in $(m-2, n-2)$, and note that $p_2'' \circ (m-2, n-1), (m-2, n) \in \mathcal{P}(M_1)$. Then,

$$\begin{aligned} p_1^+ &\prec p_2'' \circ (m-2, n-1), (m-2, n), (m-1, n), (m, n) && (2) \\ &\prec p_2' \circ (m, n-1), (m, n) && (3) \\ &\prec p_2^+, && (4) \end{aligned}$$

**Algorithm 1** The Split-and-Chop Algorithm.

```
Input   : Typical sequences $r(1), \ldots, r(m)$ and $c(1), \ldots, c(n)$
Output : A 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 $m \le 2$ then return $r(1)+c(1), r(m)+c(1), r(m)+c(2), \ldots, r(m)+c(n)$;
5      if $n \le 2$ then return $r(1)+c(1), r(1)+c(n), r(2)+c(n), \ldots, r(m)+c(n)$;
6      if $r(m-2)+c(n-1) \le r(m-1)+c(n-2)$ then return
         Chop-bottom($r[1..(m-2)], c$) $\circ (r(m-1)+c(n)), r(m)+c(n)$;
7      if $r(m-1)+c(n-2) \le r(m-2)+c(n-1)$ 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 $m \le 2$ then return $r(1)+c(1), r(1)+c(2), \ldots, r(1)+c(n), r(m)+c(n)$;
10     if $n \le 2$ then return $r(1)+c(1), r(2)+c(1), \ldots, r(m)+c(1), r(m)+c(n)$;
11     if $r(3)+c(2) \le r(2)+c(3)$ then return
         $r(1)+c(1), (r(2)+c(1)) \circ$ Chop-top($r[3..m], c$);
12     if $r(2)+c(3) \le r(3)+c(2)$ then return
         $r(1)+c(1), (r(1)+c(2)) \circ$ Chop-top($r, c[3..n]$);
```

where (2) is due to $p_1 \prec \mathcal{P}(M_1)$ and therefore $p_1 \prec p_2'' \circ (m-2, n-1), (m-2, n)$, next (3) follows from another application of the $3 \times 3$-subcase, and (4) is guaranteed since $p_2' \prec p_2$ by the iterative construction sketched above. ◀

By symmetry, these arguments also prove the "top" case of the Chop Lemmas.

▶ **Corollary 21** (Chop Lemma - Top). *Let $r$ and $c$ be typical sequences of length $m \ge 3$ and $n \ge 3$, 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 all $h \in [2]$, let $p_h \prec \mathcal{P}(M_h)$. Let $p_1^+ := (1,1),(2,1) \circ p_1$ and $p_2^+ := (1,1),(1,2) \circ p_2$.*

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

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

## 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", the details are given in Algorithm 1. Note that the base cases (lines 4, 5, 9, and 10) are easily justified: in the bottom case, the last row and column are minimum, and in the top case, the first row and column are minimum.

▶ **Lemma 22** (♣). *Let $r$ and $c$ be typical sequences of length $m$ and $n$, respectively. Then, there is an algorithm that finds in $\mathcal{O}(m+n)$ time a dominating path in the merge matrix of $r$ and $c$.*

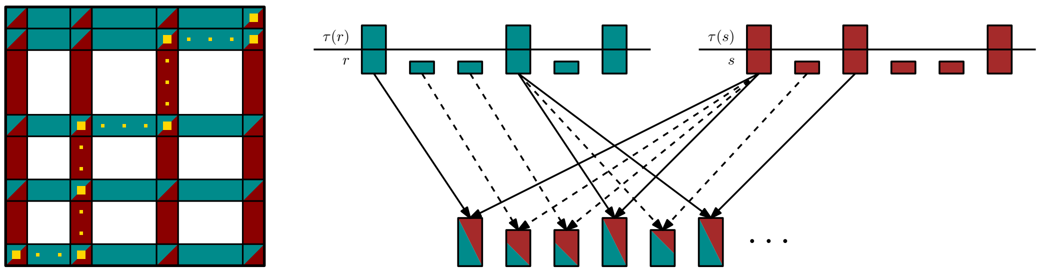

**Figure 5** Illustration of the typical lift. On the left side, the view of the merge matrix $M$, with the rows and columns corresponding to elements of the typical sequences highlighted. Inside there, $M_\tau$ can be seen as a highlighted submatrix. The merge $t'$ is depicted as the large yellow squares within $M_\tau$ and the small yellow squares outside of $M_\tau$ show its completion to the typical lift of $t$. On the right side, an illustration that does not rely on the "matrix view".

## 3.5 Generalization to Arbitrary Integer Sequences

In this section we show how to generalize Lemma 22 to arbitrary integer sequences. In particular, we will show how to construct from a merge of two typical sequences $\tau(r)$ and $\tau(s)$ that dominates all of their merges, a merge of $r$ and $s$ that dominates all merges of $r$ and $s$. The claimed result then follows from an application of Lemma 22. For an illustration of the following construction, see Figure 5.

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

**Step 1.** We construct $t' \in \tau(r) \boxplus \tau(s)$ such that $t' \prec t$ using Lemma 17. Throughout the following, consider $t'$ to be a path in the merge matrix $M_\tau$ of $\tau(r)$ and $\tau(s)$.

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

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

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

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

The following lemma captures the desired property of the typical lift, and its proof essentially follows from Lemmas 9(iii) and 17.

▶ **Lemma 23 (♣).** *Let $r$ and $s$ be integer sequences and let $q \in r \oplus s$. Let $t \in \tau(r) \oplus \tau(s)$ such that $t \prec \tau(r) \oplus \tau(s)$. Then, $\rho(t) \prec q$ (and $\rho(t) \in r \boxplus s$).*

We can now prove the main result of this work. Note that the following lemma is a strengthening of Lemma 13 in that it shows that there is always a non-diagonal dominating merge. This will be important for the algorithmic applications given in Section 4.

▶ **Lemma 24.** *Let $r$ and $c$ be integer sequences of length $m$ and $n$, respectively. There exists a dominating non-diagonal merge of $r$ and $c$, i.e. an integer sequence $t \in r \boxplus c$ such that $t \prec r \oplus c$, and this dominating merge can be computed in time $\mathcal{O}(m + n)$.*

**Proof.** The algorithm proceeds in the following steps.

**Step 1.** Compute $\tau(r)$ and $\tau(c)$.
**Step 2.** Apply the Split-and-Chop Algorithm on input $(\tau(r), \tau(c))$ to obtain $t \prec \tau(r) \oplus \tau(c)$.
**Step 3.** Return the typical lift $\rho(t)$ of $t$.

Correctness of the above algorithm follows from Corollary 16 and Lemmas 22 and 23 which together guarantee that $\rho(t) \prec r \oplus c$, and that $\rho(t)$ is a non-diagonal merge, i.e. $\rho(t) \in a \boxplus b$. By Lemma 8, Step 1 can be done in time $\mathcal{O}(m+n)$, by Lemma 22, Step 2 takes time $\mathcal{O}(m+n)$ as well, and the typical lift of $t$ can also be computed in time $\mathcal{O}(m+n)$. The overall runtime of the algorithm is $\mathcal{O}(m+n)$. ◀

## 4 Directed Width Measures of Series Parallel Digraphs

In this section, we give algorithmic consequences of the Merge Dominator Lemma. We discuss the (weighted) cutwidth on series parallel digraphs problem in Section 4.1 and (briefly) the modified cutwidth on SPD's problem in Section 4.2.

### 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.

Cutwidth of Series Parallel Digraphs

| | |
|---|---|
| *Input:* | A series parallel digraph $G$. |
| *Question:* | What is the cutwidth of $G$? |

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

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

▶ **Definition 25** (Cut-Size Sequence). *Let $G$ be a directed acyclic graph on $n$ vertices and let $\pi$ be a topological order of $G$. The sequence $x_1, \ldots, x_{n-1}$, where for $i \in [n-1]$,*

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

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

It is clear that when a cut-size sequence $\sigma(\pi_1)$ dominates another cut-size sequence $\sigma(\pi_2)$, then $\mathsf{cutw}(\pi_1) \leq \mathsf{cutw}(\pi_2)$. The proof of the next theorem goes via the following two steps. First, we show that in the dynamic programming algorithm, it is sufficient to store a dominating cut-size sequence. Second, suppose that an SPD is obtained from two smaller SPD's $G_1$ and $G_2$, and let $\pi_1$ and $\pi_2$ be topological orders of $G_1$ and $G_2$, respectively, such that $\sigma(\pi_1)$ dominates all cut-size sequences of $G_1$, and $\sigma(\pi_2)$ dominates all cut-size sequences of $G_2$. For the case $G = G_1 \vdash G_2$, we show that $\pi_1 \circ \pi_2$ yields a topological order that dominates all cut-size sequences of $G$, and for the case $G = G_1 \perp G_2$, we show that the topological order $\pi$ of $G$ such that $\sigma(\pi)$ dominates $\sigma(\pi_1) \boxplus \sigma(\pi_2)$ also dominates all cut-size sequences of $G$.

The algorithm of the following theorem in fact works for the more general problem of computing the *weighted* cutwidth of a series parallel digraph, where we are also given an arc-weight function and the value of a cut is computed as the sum of the weights of the arcs crossing the cut.

▶ **Theorem 26** (♣)**.** *Let $G$ be an SPD on $n$ vertices (together with an arc-weight function). There is an algorithm that computes in time $\mathcal{O}(n^2)$ the (weighted) cutwidth of $G$, and outputs a topological ordering that achieves the upper bound.*

## 4.2 Modified Cutwidth

We now consider the following computational problem.

Modified Cutwidth of Series Parallel Digraphs

| | |
|---|---|
| *Input:* | A series parallel digraph $G$. |
| *Question:* | What is the modified cutwidth of $G$? |

In the full version, we provide a transformation that given a series parallel digraph $G$ on $n$ vertices, outputs an instance of Weighted Cutwidth on series parallel digraphs whose digraph has $\mathcal{O}(n)$ vertices that we can use to determine the modified cutwidth of $G$. Using the algorithm for Weighted Cutwidth on series parallel digraphs due to Theorem 26, we have the following result.

▶ **Theorem 27** (♣)**.** *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.*

## 5 Conclusions

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

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

## References

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

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

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

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

5 Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 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. *J. Comput. Syst. Sci.*, 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 *Proc. 9th SODA*, pages 574–583. ACM/SIAM, 1998.

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

9 Hans L. Bodlaender and Dimitrios M. Thilikos. Constructive linear time algorithms for branchwidth. In *Proc. 24th ICALP*, volume 1256 of *LNCS*, pages 627–637. Springer, 1997.

10 Hans L. Bodlaender and Dimitrios M. Thilikos. Computing small search numbers in linear time. In *Proc. 1st IWPEC*, volume 3162 of *LNCS*, pages 37–48. Springer, 2004.

11 Mikolaj Bojanczyk and Michal Pilipczuk. Optimizing tree decompositions in MSO. In *Proc. 34th STACS*, volume 66 of *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 J. Comput.*, 38(2):629–657, 2008.

13 Martin Fürer. Faster computation of path-width. In *Proc. 27th IWOCA*, volume 9843 of *LNCS*, pages 385–396. Springer, 2016.

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

15 Jens Lagergren and Stefan Arnborg. Finding minimal forbidden minors using a finite congruence. In *Proc. 18th ICALP*, volume 510 of *LNCS*, pages 532–543. Springer, 1991.

16 Bruce A. Reed. Finding approximate separators and computing tree width quickly. In *Proc. 24th STOC*, pages 221–228. ACM, 1992.

17 Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Combin. Theory, Ser. B*, 63(1):65–110, 1995.

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

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

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