# 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

July 31, 2019

**Abstract**

In this paper, 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

*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).

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 [7]. 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 [2] 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 [12]. 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, 4, 10, 11, 12, 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 [3, 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 [7]. 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 [7] 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

[1] This approach was later used in several follow up results to obtain explicit constructive parameterized algorithms for other graph width measures, like cutwidth [20, 21], branchwidth [8], different types of search numbers like linear width [9], and directed vertex separation number [6].

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 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 [6]. The problem was shown to be NP-hard by Sethi [18] while Kessler [13] gave a $2^{\mathcal{O}(n)}$ time exact algorithm, improving over the $n^{\mathcal{O}(n)}$ naive brute-force approach. Sethi and Ullman [19] 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 [5]) 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 [5], 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 **??**, we give a number of preliminary definitions, and review existing results, including several results on typical sequences from [7]. In Section **??**, we state and prove the main technical result of this work, the Merge Dominator Lemma. Section **??** 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 2.

## 2 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 lead to new polynomial time algorithms. Since

its inception, algorithms based on typical sequences give the best asympthotic bounds for FPT algorithms for treewidth and pathwidth, as function 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] E. Amir, *Approximation algorithms for treewidth*, Algorithmica, 56 (2010), pp. 448–479.

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

[3] H. L. Bodlaender, L. Cai, J. Chen, M. R. Fellows, J. A. Telle, and D. Marx, *Open problems in parameterized and exact computation – IWPEC 2006*, Tech. Rep. UU-CS-2006-052, Department of Information and Computing Sciences, Utrecht University, 2006.

[4] H. L. Bodlaender, P. G. Drange, M. S. Dregi, F. V. Fomin, D. Lokshtanov, and M. Pilipczuk, *A $c^k n$ 5-approximation algorithm for treewidth*, SIAM Journal on Computing, 45 (2016), pp. 317–378.

[5] H. L. Bodlaender, M. R. Fellows, and D. M. Thilikos, *Derivation of algorithms for cutwidth and related graph layout parameters*, Journal of Computer and System Sciences, 75 (2009), pp. 231–244.

[6] H. L. Bodlaender, J. Gustedt, and J. A. 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, ACM/SIAM, 1998, pp. 574–583.

[7] H. L. Bodlaender and T. Kloks, *Efficient and constructive algorithms for the pathwidth and treewidth of graphs*, Journal of Algorithms, 21 (1996), pp. 358–402.

[8] H. L. Bodlaender and D. M. Thilikos, *Constructive linear time algorithms for branchwidth*, in Proceedings 24th International Colloquium on Automata, Languages and Programming, ICALP 1997, vol. 1256 of Lecture Notes in Computer Science, Springer, 1997, pp. 627–637.

[9] ——, *Computing small search numbers in linear time*, in Proceedings 1st International Workshop on Parameterized and Exact Computation, IWPEC 2004, vol. 3162 of Lecture Notes in Computer Science, Springer, 2004, pp. 37–48.

[10] M. Bojanczyk and M. Pilipczuk, *Optimizing tree decompositions in MSO*, in 34th Symposium on Theoretical Aspects of Computer Science (STACS 2017), H. Vollmer and B. Vallée, eds., vol. 66 of Leibniz International Proceedings in Informatics (LIPIcs), 2017, pp. 15:1–15:13.

[11] U. Feige, M. Hajiaghayi, and J. R. Lee, *Improved approximation algorithms for minimum weight vertex separators*, SIAM Journal on Computing, 38 (2008), pp. 629–657.

[12] M. Fürer, *Faster computation of path-width*, in Proceedings 27th International Workshop on Combinatorial Algorithms, IWOCA 2016, vol. 9843 of Lecture Notes in Computer Science (LNCS), Springer, 2016, pp. 385–396.

[13] C. W. Kessler, *Scheduling expression DAGs for minimal register need*, Computer Languages, 24 (1998), pp. 33–53.

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

[15] J. Lagergren and S. Arnborg, *Finding minimal forbidden minors using a finite congruence*, in Proceedings 18th International Colloquium on Automata, Languages and Programming, ICALP 1991, vol. 510 of Lecture Notes in Computer Science, Springer, 1991, pp. 532–543.

[16] B. A. Reed, *Finding approximate separators and computing tree width quickly*, in 24th Annual ACM Symposium on Theory of Computing (STOC 1992), ACM, 1992, pp. 221–228.

[17] N. Robertson and P. D. Seymour, *Graph minors. XIII. The disjoint paths problem*, Journal of combinatorial theory, Series B, 63 (1995), pp. 65–110.

[18] R. Sethi, *Complete register allocation problems*, SIAM Journal on Computing, 4 (1975), pp. 226–248.

[19] R. Sethi and J. D. Ullman, *The generation of optimal code for arithmetic expressions*, Journal of the ACM, 17 (1970), pp. 715–728.

[20] D. M. Thilikos, M. J. Serna, and H. L. Bodlaender, *Cutwidth I: A linear time fixed parameter algorithm*, Journal of Algorithms, 56 (2005), pp. 1–24.

[21] ——, *Cutwidth II: algorithms for partial w-trees of bounded degree*, Journal of Algorithms, 56 (2005), pp. 25–49.