On the Fine-Grained Complexity of One-Dimensional Dynamic Programming*†

Marvin Künnemann[1], Ramamohan Paturi[2], and Stefan Schneider[3]

1 University of California, San Diego, CA, USA
mkuennemann@eng.ucsd.edu
2 University of California, San Diego, CA, USA
paturi)@eng.ucsd.edu
3 University of California, San Diego, CA, USA
stschnei@eng.ucsd.edu

Abstract

In this paper, we investigate the complexity of *one-dimensional dynamic programming*, or more specifically, of the Least-Weight Subsequence (LWS) problem: Given a sequence of n data items together with weights for every pair of the items, the task is to determine a subsequence S minimizing the total weight of the pairs adjacent in S. A large number of natural problems can be formulated as LWS problems, yielding obvious $\mathcal{O}(n^2)$-time solutions.

In many interesting instances, the $\mathcal{O}(n^2)$-many weights can be succinctly represented. Yet except for near-linear time algorithms for some specific special cases, little is known about when an LWS instantiation admits a subquadratic-time algorithm and when it does not. In particular, no lower bounds for LWS instantiations have been known before. In an attempt to remedy this situation, we provide a general approach to study the fine-grained complexity of succinct instantiations of the LWS problem: Given an LWS instantiation we identify a highly parallel *core* problem that is subquadratically *equivalent*. This provides either an explanation for the apparent hardness of the problem or an avenue to find improved algorithms as the case may be.

More specifically, we prove subquadratic equivalences between the following pairs (an LWS instantiation and the corresponding core problem) of problems: a low-rank version of LWS and minimum inner product, finding the longest chain of nested boxes and vector domination, and a coin change problem which is closely related to the knapsack problem and $(\min, +)$-CONVOLUTION. Using these equivalences and known **SETH**-hardness results for some of the core problems, we deduce tight conditional lower bounds for the corresponding LWS instantiations. We also establish the $(\min, +)$-CONVOLUTION-hardness of the knapsack problem. Furthermore, we revisit some of the LWS instantiations which are known to be solvable in near-linear time and explain their easiness in terms of the easiness of the corresponding core problems.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Least-Weight Subsequence, SETH, Fine-Grained Complexity, Knapsack, Subquadratic Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.21

* A full version is available at [31].

† This research is supported by the Simons Foundation. This research is supported by NSF grant CCF-1213151 from the Division of Computing and Communication Foundations. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 21; pp. 21:1–21:15
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Dynamic programming (DP) is one of the most fundamental paradigms for designing algorithms and a standard topic in textbooks on algorithms. Scientists from various disciplines have developed DP formulations for basic problems encountered in their applications. However, it is not clear whether the existing (often simple and straightforward) DP formulations are in fact optimal or nearly optimal. Our lack of understanding of the optimality of the DP formulations is particularly unsatisfactory since many of these problems are computational primitives.

Interestingly, there have been recent developments regarding the optimality of standard DP formulations for some specific problems, most importantly, conditional lower bounds assuming the Strong Exponential Time Hypothesis (**SETH**) [26]. Let us consider the longest common subsequence (LCS) problem as an illustrative example. It is defined as follows: Given two strings x and y of length at most n, compute the length of the longest string z that is a subsequence of both x and y. The standard DP formulation for the LCS problem involves computing a two-dimensional table requiring $\mathcal{O}(n^2)$ steps. This algorithm is slower than the fastest known algorithm due to Masek and Paterson [33] only by a polylogarithmic factor. However, there has been no progress in finding more efficient algorithms for this problem since the 1980s, which prompted attempts as early as in 1976 [5] to understand the barriers for efficient algorithms and to prove lower bounds. Unfortunately, there have not been any nontrivial unconditional lower bounds for this or any other problem in general models of computation. This state of affairs prompted researchers to consider *conditional* lower bounds based on conjectures such as 3-Sum conjecture [18] and more recently based on **ETH** [27] and **SETH** [26]. Researchers have found **ETH** and **SETH** to be useful to explain the exact complexity of several **NP**-complete problems (see the survey paper [32]). Surprisingly, Ryan Williams [39] has found a simple reduction from the CNF-SAT problem to the orthogonal vectors problem which under **SETH** leads to a matching quadratic lower bound for the orthogonal vectors problem. This in turn led to a number of conditional lower bound results for problems in **P** (including LCS and related problems) under **SETH** [6, 1, 10, 2, 22]. Also see [37] for a recent survey.

The DP formulation of the LCS problem is perhaps the conceptually simplest example of a *two-dimensional* DP formulation. In the standard formulation, each entry of an $n \times n$ table is computed in constant time. This property is typical for *alignment problems* which, for example, are used to model similarity between gene or protein sequences and for which LCS and Edit distance are the most prominent examples. Tight conditional lower bounds have recently been proved for a number of alignment problems [8, 6, 1, 10, 3].

In contrast, there are many problems for which natural quadratic-time DP formulations compute a *one-dimensional* table of length n by spending $\mathcal{O}(n)$-time per entry. The question arises: Can similar optimality results as for alignment problems be obtained for this fundamentally different setting? In pursuit of an answer, we investigate the optimality of one-dimensional DP formulations and obtain new (conditional) lower bounds which match the complexity of these standard DP formulations.

1-dimensional DP: The Least-Weight Subsequence (LWS) Problem. In this paper, we investigate the optimality of the standard DP formulation of the LWS problem. A classic example of an LWS problem is airplane refueling [24]: Given airport locations on a line, and a preferred distance per hop k (in miles), we define the penalty for flying k' miles as $(k - k')^2$. The goal is then to find a sequence of airports terminating at the last airport that minimizes the sum of the penalties. We now define the LWS problem formally.

▶ **Problem 1.1** (LWS). *We are given weights $w_{i,j} \in \{-W, \dots, W\} \cup \{\infty\}$ for every pair $0 \le i < j \le n$ and an arbitrary function $g : \mathbb{Z} \to \mathbb{Z}$. The* LWS *problem is to determine $F[n]$ which is defined by the following DP formulation.*

$$
\begin{aligned}
&F[0] = 0, \\
&F[j] = \min_{0 \le i < j} g(F[i]) + w_{i,j} \qquad \text{for } j = 1, \dots, n. \qquad (1)
\end{aligned}
$$

In the above definition, we did not specify the precise encoding of the problem. We typically consider *succinct instantiations* of LWS, where the input has subquadratic size (typically $\tilde{\mathcal{O}}(n)$) and the weights are a function of the input. In many cases, the input is a list of data items $x_0, \dots, x_n$ and $w_{i,j}$ is a function of x_i and x_j. For example, to formulate airplane refueling as an LWS problem, we let x_i be the location of the i'th airport, g be the identity function, and $w_{i,j} = (x_j - x_i - k)^2$.

The generality of the LWS definition captures a large variety of problems: it not only encompasses classical problems such as the pretty printing problem due to Knuth and Plass [30], the airplane refueling problem [24] and the longest increasing subsequence (LIS) [17], but also the unbounded subset sum problem [36, 9], a more general coin change problem that is effectively equivalent to the unbounded knapsack problem, 1-dimensional k-means clustering problem [23], finding longest R-chains (for an arbitrary binary relation R), and many others (for a more detailed overview and problem definitions, see the full version [31]).

Under mild assumptions on the encoding of the data items and weights, any instantiation of the LWS problems can be solved in time $\mathcal{O}(n^2)$ using (1) for determining the values $F[j], j = 1, \dots, n$ in time $\mathcal{O}(n)$ each. However, the best known algorithms for the LWS problems differ quite significantly in their time complexity. Some problems including the pretty printing problem, the airline refueling problem and LIS turn out to be solvable in near-linear time, while no subquadratic algorithms are known for the unbounded knapsack problem or for finding the longest R-chain.

The main goal of the paper is to investigate the optimality of the LWS DP formulation for various problems by proving conditional lower bounds.

Succinct LWS instantiations. In the extremely long presentation of an LWS problem, the weights $w_{i,j}$ are given explicitly. This is, however, not a very interesting case from a computational point of view, as the standard DP formulation takes linear time (in the size of the input) to compute $F[n]$. In the example of the airplane refueling problem, the size of the input is only $\mathcal{O}(n)$ assuming that the values of the data items are bounded by some polynomial in n. For such succinct representations, we ask if the quadratic-time algorithm based on the standard LWS DP formulation is optimal. Our approach is to study several natural succinct versions of the LWS problem (by specifying the type of data items and the weight function[1]) and determine their complexity.

Our Contributions and Results. The main contributions of our paper include a general framework for reducing succinct LWS instantiations to what we call the *core* problems and proving subquadratic equivalences between them. Such subquadratic equivalences are interesting for two reasons. First, they allow us to conclude conditional lower bounds for certain LWS instantiations, where previously no lower bounds are known. Second,

[1] In all our applications, the function g is the trivial identity function.

subquadratic (or more general fine-grained) equivalences are more useful since they let us translate *easiness* in addition to hardness results.

Our results include tight (up to subpolynomial factors) conditional lower bounds for several LWS instantiations with succinct representations. These instantiations include the coin change problem, low-rank versions of the LWS problem, and the longest subchain problems. Our results are somewhat more general. We propose a *factorization* of the LWS problem into a *core* problem and a fine-grained reduction from the LWS problem to the core problem. The idea is that core problems (which are often well-known problems) capture the hardness of the LWS problem and act as a potential barrier for more efficient algorithms. While we do not formally define the notion of a core problem, we identify several core problems which share several interesting properties. For example, they do not admit natural DP formulations and are easy to parallelize. In contrast, the quadratic-time DP formulation of LWS problems requires the entries $F[i]$ to be computed in order, suggesting that the general problem might be inherently sequential.

The reductions between LWS problems and core problems involve a natural intermediate problem, which we call the STATIC-LWS problem. We first reduce the LWS problem to the STATIC-LWS problem in a general way and then reduce the STATIC-LWS problem to a core problem. The first reduction is divide-and-conquer in nature and is inherently sequential. The latter reduction is specific to the instantiation of the LWS problem. The STATIC-LWS problem is easy to parallelize and does not have a natural DP formulation. However, the problem is not necessarily a natural problem. The STATIC-LWS problem can be thought of as a generic core problem, but it is output-intensive.

In the other direction, we show that many of the core problems can be reduced to the corresponding LWS instantiations thus establishing an equivalency between LWS instantiations and their core problems. This equivalence enables us to translate both the hardness and easiness results (i.e., the subquadratic-time algorithms) for the core problems to the corresponding LWS instantiations.

The first natural succinct representation of the LWS problem we consider is the low-rank LWS problem, where the weight matrix $\mathbf{W} = (w_{i,j})$ is of low rank and thus representable as $\mathbf{W} = L \cdot R$ where L and R^{T} are $(n \times n^{o(1)})$-matrices. For this low-rank LWS problem, we identify the minimum inner product problem (MININNPROD) as a suitable core problem. It is only natural and not particularly surprising that MININNPROD can be reduced to the low-rank LWS problem which shows the **SETH**-hardness of the low-rank LWS problem. The other direction is more surprising: Inspired by an elegant trick of Vassilevska Williams and Williams [40], we are able to show a subquadratic-time reduction from the (highly sequential) low-rank LWS problem to the (highly parallel) MININNPROD problem. Thus, the very compact problem MININNPROD problem captures exactly the complexity of the low-rank LWS problem (under subquadratic reductions).

We also show that the coin change problem is subquadratically equivalent to the $(\min,+)$-CONVOLUTION problem. In the coin change problem, the weight matrix $\mathbf{W}$ is succinctly given as a Toeplitz matrix. At this point, the conditional hardness of the $(\min,+)$-CONVOLUTION problem is unknown. Only very recently and independent of our work, a detailed treatment of Cygan et al. [13] considers quadratic-complexity of $(\min,+)$-CONVOLUTION as a hardness assumption and discusses its relation to more established assumptions. The quadratic-time hardness of the $(\min,+)$-CONVOLUTION problem would be very interesting, since it is known that the $(\min,+)$-CONVOLUTION problem is reducible to the 3-Sum problem and the APSP problem (see also [13]). However, recent results give surprising subquadratic-time algorithms for special cases of $(\min,+)$-CONVOLUTION [12]. If these subquadratic-time

Table 1 Summary of our results.

Name	Weights	Equivalent Core	Reference
Coin Change	Toeplitz matrix:	$(\min,+)$-CONVOLUTION	Theorem 5.9
	$w_{i,j} = w_{j-i}$		
	Remark: Subquadratically equivalent to UNBOUNDEDKNAPSACK		
LowRankLWS	Low rank representation:	MININNPROD	Theorem 4.7
	$w_{i,j} = \langle \sigma_i, \mu_j \rangle$		
R-chains	matrix induced by R:	SELECTION(R)	Theorem 6.3
	$w_{i,j} = w_j$ if $R(x_i, x_j)$ and ∞ o/w		Theorem 6.4
	Remark: Results below are corollaries.		
NestedBoxes	$w_{i,j} = -1$ if B_j contains B_i	VECTORDOMINATION	
SubsetChain	$w_{i,j} = -1$ if $S_i \subseteq S_j$	ORTHOGONALVECTORS	

algorithms extend to the general $(\min,+)$-CONVOLUTION problem, our equivalence result also provides a subquadratic-time algorithm for the coin change problem and the closely related unbounded knapsack problem. Our reductions also give, as a corollary, a quadratic-time $(\min,+)$-CONVOLUTION-based lower bound for the bounded case of knapsack. We remark that independently of our results, [13] gave randomized subquadratic equivalences of $(\min,+)$-CONVOLUTION to unbounded knapsack (while we give deterministic reductions) and bounded Knapsack (where we only give a $(\min,+)$-CONVOLUTION-based lower bound).

We next consider the problem of finding longest chains: here, we search for the longest subsequence (chain) in the input sequence such that all adjacent pairs in the subsequence are contained in some binary relation R. We show that for any binary relation R satisfying certain conditions the chaining problem is subquadratically equivalent to a corresponding (highly parallel) selection problem. As corollaries, we get equivalences between finding the longest chain of nested boxes (NESTEDBOXES) and VECTORDOMINATION as well as between finding the longest subset chain (SUBSETCHAIN) and the orthogonal vectors (OV) problem. Interestingly, these results have algorithmic implications: known algorithms for low-dimensional vector domination and low-dimensional orthogonal vectors translate to faster algorithms for low-dimensional NESTEDBOXES and SUBSETCHAIN for small universe size.

Table 1 lists the LWS succinct instantiations (as discussed above) and their corresponding core problems. For a detailed treatment of all LWS instantiations and core problems considered in this work, see the full version of this paper [31].

Finally, we revisit classic problems including the longest increasing subsequence problem, the unbounded subset sum problem and the concave LWS problem and analyze the STATIC-LWS instantiations to immediately infer that the corresponding core problem can be solved in near-linear time. Table 2 gives an overview of some of the problems we look at in this context.

Related Work. LWS has been introduced by Hirschberg and Larmore [24]. If the weight function satisfies the *quadrangle inequality* formalized by Yao [41], one obtains the *concave LWS* problem (CONCLWS), for which they give an $\mathcal{O}(n \log n)$-time algorithm. Subsequently, improved algorithms solving CONCLWS in time $\mathcal{O}(n)$ were given [38, 20]. This yields a fairly large class of weight functions (including, e.g., the pretty printing and airplane refueling problems) for which linear-time solutions exist. To generalize this class of problems, further

Table 2 Near-linear time algorithms following from the proposed framework.

Name	**Weights**	$\tilde{\mathcal{O}}(n)$**-algorithm via**	**Reference**
Longest Increasing Subsequence	matrix induced by $R_<$: $w_{i,j} = -1$ if $x_i < x_j$	Sorting	[17], full version [31]
Unbounded Subset Sum	Toeplitz $\{0, \infty\}$ matrix: $w_{i,j} = w_{j-i} \in \{0, \infty\}$	Convolution	[9], full version [31]
Concave 1-dim. DP	concave matrix: $w_{i,j} + w_{i',j'} \leq w_{i',j} + w_{i,j'}$ for $i \leq i' \leq j \leq j'$	SMAWK problem	[24, 20, 38], full version [31]

works address convex weight functions[2] [19, 35, 29] as well as certain combinations of convex and concave weight functions [15] and provide near-linear time algorithms. For a more comprehensive overview over these algorithms and further applications of the LWS problem, we refer the reader to Eppstein's PhD thesis [16].

Apart from these notions of concavity and convexity, results on succinct LWS problems are typically more scattered and problem-specific (see, e.g., [17, 30, 9, 23]; furthermore, a closely related recurrence to (1) pops up when solving bitonic TSP [14]). An exception to this rule is a study of the parallel complexity of LWS [21].

Organization. After setting up notation and conventions in Section 2, Section 3 gives a general reduction from LWS instantiations to Static-LWS that is independent of the representation of the weight matrix. Section 4 contains the result on low-rank LWS. Section 5 proves the subquadratic equivalence of the coin change problem and $(\min, +)$-convolution, while Section 6 discusses chaining problems and their corresponding selection (core) problem. Due to space constraints, most proofs and our discussion of near-linear time algorithms are deferred to the full version of this article [31].

2 Preliminaries

In this section, we state our notational conventions and list the main problems considered in this work.

Notation and Conventions. Problem A *subquadratically reduces* to problem B, denoted $A \leq_2 B$, if for any $\varepsilon > 0$ there is a $\delta > 0$ such that the existence of a $\mathcal{O}(n^{2-\varepsilon})$-time algorithm for B implies a $\mathcal{O}(n^{2-\delta})$-time algorithm for A. We call the two problems subquadratically equivalent, denoted $A \equiv_2 B$, if there are subquadratic reductions both ways.

We let $[n] := \{1, \ldots, n\}$. When stating running time, we use the notation $\tilde{\mathcal{O}}(\cdot)$ to hide polylogarithmic factors. For a problem P, we write T^P for its time complexity. We generally assume the word-RAM model of computation with word size $w = \Theta(\log n)$. For most problems defined in this paper, we consider inputs to be integers in the range $\{-W, \ldots, W\}$ where W fits in a constant number of words[3]. For vectors, we use d for the dimension and generally assume $d = n^{o(1)}$.

[2] A weight function is convex if it satisfies the inverse of the quadrangle inequality.

[3] For the purposes of our reductions, even values up to $W = 2^{n^{o(1)}}$ would be fine.

Succinct LWS Instantiations. In the definition of LWS (Problem 1.1) we did not fix the encoding of the problem (in particular the representation of the weights $w_{i,j}$ and the function g). Assuming that g and the weights can be determined in $\tilde{\mathcal{O}}(1)$ and that $W = \mathtt{poly}(n)$, this problem can naturally be solved in time $\tilde{\mathcal{O}}(n^2)$, by evaluating the central recurrence (1) for each $j = 1, \ldots, n$ – this takes $\tilde{\mathcal{O}}(n)$ time for each j, since we take the minimum over at most n expressions that can be evaluated in time $\tilde{\mathcal{O}}(1)$ by accessing the previously computed entries $F[0], \ldots, F[j-1]$ as well as computing g. We assume from now on that g is the identity function, as this is the case for all our applications. Thus it suffices to define the type of data items and the corresponding weight matrix to specify an LWS instantiation. Throughout this paper, whenever we fix a representation of the weight matrix $\mathbf{W} = (w_{i,j})_{i,j}$, we denote the corresponding problem LWS($\mathbf{W}$).

3 Static LWS

Our reductions from LWS instantiations to core problems go through intermediate problems that share some of the characteristics of core problems, as well as some of the characteristics of LWS. In particular, these problems are naturally parallelizable and their brute-force algorithm is already quadratic time, similar to core problems. On the other hand their definitions are closely related to the definition of LWS. Other than core problems, our intermediate problems are not decision problems but ask to compute some linear sized output. Towards making this notion more precise, we define a generic intermediate problem called Static-LWS.

▶ **Problem 3.1** (Static-LWS($\mathbf{W}$)). *Fix an instance of* LWS($\mathbf{W}$). *Given intervals of indices $I := \{a+1, \ldots, a+N\}$ and $J := \{a+N+1, \ldots, a+2N\}$ with a, N such that $I, J \subseteq [n]$, together with the values $F[a+1], \ldots, F[a+N]$, the Static Least-Weight Subsequence Problem (*Static-LWS*) asks to determine*

$$F'[j] := \min_{i \in I} F[i] + w_{i,j} \qquad \text{for all } j \in J.$$

The main purpose of this section is to give a reduction from LWS($\mathbf{W}$) to Static-LWS($\mathbf{W}$) that is independent of the weight matrix $\mathbf{W}$ and therefore independent of the succinct LWS instantiations we consider throughout this paper. This reduction is a key step in our reductions from LWS to their corresponding core problems.

The reduction is a divide-and-conquer scheme that divides the LWS problem into two subproblems of half the size each and Static-LWS to combine the two. Crucially, the two subproblems have to be solved sequentially. The reduction therefore captures the sequential nature of the LWS problem, while Static-LWS captures a parallelizable part of the problem.

In a certain sense, this reduction has appeared implicitly in previous work on LWS [24]. In particular, the reduction of ConcLWS to the SMAWK problem by Galil and Park [20] can be thought of as a variant of this reduction specialized to the concave case to avoid log-factors.

▶ **Lemma 3.2** (LWS($\mathbf{W}$) $\leq_2$ Static-LWS($\mathbf{W}$)). *For any choice of $\mathbf{W}$, if* Static-LWS($\mathbf{W}$) *can be solved in time $\mathcal{O}(N^{2-\varepsilon})$ for some $\varepsilon > 0$, then* LWS($\mathbf{W}$) *can be solved in time $\tilde{\mathcal{O}}(n^{2-\varepsilon})$.*

Proof. In what follows, we fix LWS as LWS($\mathbf{W}$) and Static-LWS as Static-LWS($\mathbf{W}$).

We define the subproblem $S(\{i, \ldots, j\}, (f_i, \ldots, f_j))$ that given an interval spanned by $1 \leq i \leq j \leq n$ and values $f_k = \min_{0 \leq k' < i} F[k'] + w_{k',k}$ for each point $k \in \{i, \ldots, j\}$, computes all values $F[k]$ for $k \in \{i, \ldots, j\}$. Note that a call to $S([n], (w_{0,1}, \ldots, w_{0,n}))$ solves the LWS problem, since $F[0] = 0$ and thus the values of $f_k, k \in [n]$ are correctly initialized.

Algorithm 1 Reducing LWS to Static-LWS

1: **function** $S(\{i,\dots,j\},(f_i,\dots,f_j))$
2:     **if** $i=j$ **then**
3:         **return** $F[i]\leftarrow f_i$
4:     $m\leftarrow\lceil\frac{j-i}{2}\rceil$
5:     $(F[i],\dots,F[i+m-1])\leftarrow S(\{i,\dots,i+m-1\},(f_i,\dots,f_{i+m-1}))$
6:     solve Static-LWS on the subinstance given by $I:=\{i,\dots,i+m-1\}$ and $J:=\{i+m,\dots,i+2m-1\}$:
7:         ▷ obtains values $F'[k]=\min_{i\le k'<i+m}F[k']+w_{k',k}$ for $k=i+m,\dots,i+2m-1$.
8:     $f'_k\leftarrow\min\{f_k,F'[k]\}$ for all $k=i+m,\dots,i+2m-1$.
9:     $(F[i+m],\dots,F[i+2m-1])\leftarrow S(\{i+m,\dots,i+2m-1\},(f'_{i+m},\dots,f'_{i+2m-1}))$
10:     **if** $j=i+2m$ **then**
11:         $F[j]:=\min\{f_j,\min_{i\le k<j}F[k]+w_{k,j}\}$.
12:     **return** $(F[i],\dots,F[j])$

We solve S using Algorithm 1.

We briefly argue correctness, using the invariant that $f_k=\min_{0\le k'<i}F[k']+w_{k',k}$ in every call to S. If S is called with $i=j$, then the invariant yields $f_i=\min_{0\le k'<i}F[k']+w_{k',i}=F[i]$, thus $F[i]$ is computed correctly. For the call in Line 5, the invariant is fulfilled by assumption, hence the values $(F[i],\dots,F[i+m-1])$ are correctly computed. For the call in Line 9, we note that for $k=i+m,\dots,i+2m-1$, we have that f'_k equals

$$\min\{f_k,F'[k]\}=\min\{\min_{0\le k'<i}F[k']+w_{k',k},\min_{i\le k'<i+m}F[k']+w_{k',k}\}=\min_{0\le k'<i+m}F[k']+w_{k',k}.$$

Hence the invariant remains satisfied. Thus, the values $(F[i+m],\dots,F[i+2m-1])$ are correctly computed. Finally, if $j=i+2m$, we compute the remaining value $F[j]$ correctly, since $f_j=\min_{0\le k<i}F[k]+w_{k,j}$ by assumption.

To analyze the running time $T^S(n)$ of S on an interval of length $n:=j-i+1$, note that each call results in two recursive calls of interval lengths at most $n/2$. In each call, we need an additional overhead that is linear in n and $T^{\text{Static-LWS}}(n/2)$. Solving the corresponding recursion $T^S(n)\le 2T^S(n/2)+T^{\text{Static-LWS}}(n/2)+\mathcal{O}(n)$, we obtain that an $\mathcal{O}(N^{2-\varepsilon})$-time algorithm Static-LWS, with $0<\varepsilon<1$ yields $T^{\text{LWS}}(n)\le T^S(n)=\mathcal{O}(n^{2-\varepsilon})$. Similarly, an $\mathcal{O}(N\log^c N)$-time algorithm for Static-LWS would result in an $\mathcal{O}(n\log^{c+1}n)$-time algorithm for LWS. ◀

4 LowRankLWS

In this section we prove the first equivalence between an instantiation of LWS and a core problem. Specifically, we first analyze the following canonical succinct representation of a low-rank weight matrix $\mathbf{W}=(w_{i,j})_{i,j}$: If $\mathbf{W}$ is of rank $d\ll n$, we can write it more succinctly as $\mathbf{W}=L\cdot R$, where L and R are $(n\times d)$- and $(d\times n)$ matrices, respectively. We can express the resulting natural LWS problem equivalently as follows.

▶ **Problem 4.1** (LowRankLWS). *We define the* LWS *instantiation* LowRankLWS $=$ LWS($\mathbf{W}_{\text{LowRank}}$) *as follows.*
Data: *out-vectors* $\mu_0,\dots,\mu_{n-1}\in\{-W,\dots,W\}^d$*, in-vectors* $\sigma_1,\dots,\sigma_n\in\{-W,\dots,W\}^d$
Weights: $w(i,j)=\langle\mu_i,\sigma_j\rangle$ *for* $0\le i<j\le n$

In this section, we show that this problem is equivalent, under subquadratic reductions, to the following *non-sequential* problem.

▶ **Problem 4.2** (MinInnProd). *Given $a_1, \ldots, a_n, b_1, \ldots, b_n \in \{-W, \ldots, W\}^d$ and a natural number $r \in \mathbb{Z}$, determine if there is a pair i, j satisfying $\langle a_i, b_j \rangle \leq r$.*

This is interesting for a number of reasons. For one, MinInnProd is a fairly natural problem and, as opposed to LowRankLWS it is not inherently sequential in its definition. We understand MinInnProd comparably well both from an upper and from a lower bound perspective. Using ray shooting data structures [34] we can solve MinInnProd in strongly subquadratic time if d is constant. At the same time, if $d = \omega(\log n)$, the problem is quadratic-time **SETH**-hard. By showing subquadratic equivalence between MinInnProd and LowRankLWS, we can conclude both these results, as well as any future improvements, for LowRankLWS.

There is a simple reduction from MinInnProd to LowRankLWS that along the way proves quadratic-time **SETH**-hardness of LowRankLWS.

▶ **Lemma 4.3.** *It holds that $T^{\text{MinInnProd}}(n, d, W) \leq T^{\text{LowRankLWS}}(2n+1, d+2, dW) + \mathcal{O}(nd)$.*

To prove the other direction, we will use the quite general approach to compute the sequential LWS problem by reducing to Static-LWS (Lemma 3.2). In particular, for the special case of LowRankLWS, it is not difficult to see that its static version boils down to the following natural reformulation.

▶ **Problem 4.4** (AllInnProd). *Given vectors $a_1, \ldots, a_n \in \{-W, \ldots, W\}^d$ and $b_1, \ldots, b_n \in \{-W, \ldots, W\}^d$, determine* for all *$j \in [n]$, the value $\min_{i \in [n]} \langle a_i, b_j \rangle$.*

▶ **Lemma 4.5** (Static-LWS($\mathbf{W}_{\text{LowRank}}$) $\leq_2$ AllInnProd). *We have*

$$T^{\text{Static-LWS}(\mathbf{W}_{\text{LowRank}})}(n, d, W) \leq T^{\text{AllInnProd}}(n, d+1, nW) + \mathcal{O}(nd).$$

Finally, inspired by an elegant trick of [40], we reduce AllInnProd to MinInnProd.

▶ **Lemma 4.6** (AllInnProd $\leq_2$ MinInnProd). *We have*

$$T^{\text{AllInnProd}}(n, d, W) \leq \mathcal{O}(n \cdot T^{\text{MinInnProd}}(\sqrt{n}, d+3, ndW^2) \cdot \log^2 nW).$$

By the sequence of lemmas above and Lemma 3.2, we obtain our subquadratic equivalence of LowRankLWS to its core problem.

▶ **Theorem 4.7.** *We have* LowRankLWS $\equiv_2$ MinInnProd.

5 Coin Change and Knapsack Problems

In this section, we focus on the following problem related to Knapsack: Assume we are given coins of denominations $d_1, \ldots, d_m$ with corresponding weights $w_1, \ldots, w_m$ and a target value n, determine a way to represent n using these coins (where each coin can be used arbitrarily often) minimizing the total sum of weights of the coins used. Since without loss of generality $d_i \leq n$ for all i, we can assume that $m \leq n$ and think of n as our problem size. In particular, we describe the input by weights $w_1, \ldots, w_n$ where w_i denotes the weight of the coin of denomination i (if no coin with denomination i exists, we set $w_i = \infty$). It is straightforward to see that this problem is an LWS instance LWS($\mathbf{W}_{\text{cc}}$), where the weight matrix $\mathbf{W}_{\text{cc}}$ is a Toeplitz matrix.

▶ **Problem 5.1** (CC). *We define the following* LWS *instantiation* $\mathrm{CC} = \mathrm{LWS}(\mathbf{W}_{\mathrm{cc}})$.
Data: *weight sequence* $w = (w_1, \ldots, w_n)$ *with* $w_i \in \{-W, \ldots, W\} \cup \{\infty\}$
Weights: $w_{i,j} = w_{j-i}$ *for* $0 \leq i < j \leq n$

Translated into a Knapsack-type formulation (i.e., denominations are weights, weights are profits, and the objective becomes to maximize the profit), the problem differs from UnboundedKnapsack only in that it searches for the most profitable multiset of items of weight *exactly* n, instead of *at most* n.

▶ **Problem 5.2** (UnboundedKnapsack). *We are given a sequence of profits* $p = (p_1, \ldots, p_n)$ *with* $p_i \in \{0, 1, \ldots, W\}$*, that is, the item of size* i *has profit* p_i*. Find the total profit of the multiset of indices* I *such that* $\sum_{i \in I} i \leq n$ *and the total profit* $\sum_{i \in I} p_i$ *is maximized.*

The purpose of this section is to show that both CC and UnboundedKnapsack are subquadratically equivalent to the $(\min, +)$-convolution problem. Along the way, we also prove quadratic-time $(\min, +)$-convolution-hardness of Knapsack. Recall the definition of $(\min, +)$-convolution.

▶ **Problem 5.3** ($(\min, +)$-convolution). *Given* n*-dimensional vectors* $a = (a_0, \ldots, a_{n-1})$, $b = (b_0, \ldots, b_{n-1}) \in \{-W, \ldots, W\}^n$*, determine its* $(\min, +)$-convolution $a * b$ *defined by*

$$(a * b)_k = \min_{0 \leq i,j < n : i+j=k} a_i + b_j \qquad \text{for all } 0 \leq k \leq 2n-2.$$

As opposed to the classical convolution, solvable in time $\mathcal{O}(n \log n)$ using FFT, no strongly subquadratic algorithm for $(\min, +)$-convolution is known. Compared to the popular orthogonal vectors problem, we have less support for believing that no $\mathcal{O}(n^{2-\varepsilon})$-time algorithm for $(\min, +)$-convolution exists. In particular, interesting special cases can be solved in subquadratic time [12] and there are subquadratic-time co-nondeterministic and nondeterministic algorithms [7, 11]. At the same time, breaking this long-standing quadratic-time barrier is a prerequisite for progress on refuting the 3SUM and APSP conjectures (see also [13]). This makes it an interesting target particularly for proving subquadratic *equivalences*, since both positive and negative resolutions of this open question appear to be reasonable possibilities.

To obtain our result, we address two issues: (1) We show an equivalence between the problem of determining only the value $F[n]$, i.e., the best way to give change only for the target value n, and to determine *all values* $F[1], \ldots, F[n]$, which we call the *output-intensive version*. (2) We show that the output-intensive version is subquadratic equivalent to $(\min, +)$-convolution.

▶ **Problem 5.4** (oiCC). *The output-intensive version of* CC *is to determine, given an input to* CC*, all values* $F[1], \ldots, F[n]$.

We first consider issue (2) and prove $(\min, +)$-convolution-hardness of oiCC.

▶ **Lemma 5.5** ($(\min, +)\textsc{conv} \leq_2 \textsc{oiCC}$). *We have* $T^{(\min,+)\textsc{conv}}(n, W) \leq T^{\textsc{oiCC}}(6n, 4(2W + 1)) + \mathcal{O}(n)$.

Using the notion of Static-LWS, the other direction is straight-forward.

▶ **Lemma 5.6.** *We have* $\textsc{oiCC} \leq_2 \textsc{Static-LWS}(\mathbf{W}_{\mathrm{cc}}) \leq_2 (\min, +)\textsc{conv}$.

The last two lemmas resolve issue (2). We proceed to issue (1) and show that the output-intensive version is subquadratically equivalent to both CC and UnboundedKnapsack that only ask to determine a single output number.

It is trivial to see that UnboundedKnapsack $\leq_2$ oiCC. Furthermore, there is a simple reduction from CC to UnboundedKnapsack.

▶ **Oberservation 5.7** (CC $\leq_2$ UnboundedKnapsack $\leq_2$ oiCC)**.** *We have* $T^{\mathrm{CC}}(n, W) \leq T^{\mathrm{UnboundedKnapsack}}(n, nW) + \mathcal{O}(n)$ *and* $T^{\mathrm{UnboundedKnapsack}}(n, W) \leq T^{\mathrm{oiCC}}(n, W) + \mathcal{O}(n)$*.*

The remaining part is similar in spirit to Lemma 4.6: Somewhat surprisingly, the same general approach works despite the much more sequential nature of Knapsack and CC – this sequentiality can be taken care of by a more careful treatment of appropriate subproblems that involves solving them in a particular order and feeding them with information gained during the process.

▶ **Lemma 5.8** (oiCC $\leq_2$ CC)**.** *We have* $T^{\mathrm{oiCC}}(n, W) \leq \mathcal{O}(\log(nW) \cdot n \cdot T^{\mathrm{CC}}(24\sqrt{n}, 3n^2W))$*.*

The lemmas above and their underlying reductions prove the following theorem.

▶ **Theorem 5.9.** *We have* $(\min, +)$conv $\equiv_2$ CC $\equiv_2$ UnboundedKnapsack*. Furthermore, the bounded version of* Knapsack *admits no strongly subquadratic-time algorithm unless* $(\min, +)$-convolution *can be solved in strongly subquadratic time.*

6 Chain LWS

In this section we consider a special case of Least-Weight Subsequence problems called the Chain Least-Weight Subsequence (ChainLWS) problem. This captures problems in which edge weights are given implicitly by a relation R that determines which pairs of data items we are allowed to chain. The aim is to find the longest chain.

An example of a Chain Least-Weight Subsequence problem is the NestedBoxes problem. Given n boxes in d dimensions, given as non-negative, d-dimensional vectors $b_1, \ldots, b_n$, find the longest chain such that each box fits into the next (without rotation). We say box that box a fits into box b if for all dimensions $1 \leq i \leq d$, $a_i \leq b_i$.

NestedBoxes is not immediately a Least-Weight Subsequence problem, as for Least-Weight subsequence problems we are given a sequence of data items, and require any sequence to start at the first item and end at the last. However, we can easily convert NestedBoxes into a LWS problem by sorting the vectors by the sum of the entries and introducing two special boxes, one very small box $\perp$ such that $\perp$ fits into any box b_i and one very large box $\top$ such that any b_i fits into $\top$.

We define the Chain Least-Weight Subsequence problem with respect to any relation R and consider a weighted version where data items are given weights. To make the definition consistent with the definition of LWS the output is the weight of the sequence that minimizes the sum of the weights.

▶ **Problem 6.1** (ChainLWS)**.** *Fix a set of objects* D *and a relation* $R \subseteq D \times D$*. We define the following* LWS *instantiation* ChainLWS(R) = LWS$(\mathbf{W}_{\mathrm{ChainLWS}(R)})$*.*
Data: *sequence of objects* $d_0, \ldots, d_n \in D$ *with weights* $w_1, \ldots, w_n \in \{-W, \ldots, W\}$*.*
Weights: $w_{i,j} = \begin{cases} w_j & \text{if } (x_i, x_j) \in R, \\ \infty & \text{otherwise,} \end{cases}$ *for* $0 \leq i < j \leq n$*.*

The input to the (weighted) Chain Least-Weight Subsequence problem is a *sequence* of data items, and not a set. Finding the longest chain in a *set* of data items is **NP**-complete in general. For example, consider the box overlap problem: The input is a set of boxes in two dimensions, given by the top left corner and the bottom right corner, and the relation consists of all pairs such that the two boxes overlap. This problem is a generalization of the Hamiltonian path problem on induced subgraphs of the two-dimensional grid, which is an **NP**-complete problem [28].

We relate ChainLWS(R) to the class of *selection* problems with respect to the same relation R.

▶ **Problem 6.2** (Selection Problem). *Let D be a set of objects, let $R \subseteq D \times D$ be a relation and let $D_1, D_2 \subseteq D^n$. Given two sequences of inputs $(a_1, \dots, a_n) \in D_1$ and $(b_1, \dots, b_n) \in D_2$, determine if there is i, j satisfying $R(a_i, b_j)$. We denote this selection problem with respect to the relation R and sets D_1, D_2 by* Selection(R^{D_1,D_2})*. If $D_1 = D_2 = D^n$, we denote the problem by* Selection(R)*.*

The class of selection problems includes several well-studied problems including MinInnProd, OV [39, 4] and VectorDomination [25].

We give a subquadratic reduction from ChainLWS(R) to Selection(R), independently of R. The proof is again based on Static-LWS and a variation on a trick of [40].

▶ **Theorem 6.3.** *For all relations R such that R can be computed in time subpolynomial in the number of data items n,* ChainLWS$(R) \leq_2$ Selection(R)*.*

For the other direction, we do not have a reduction that is independent of the relation R. Instead, we give sufficient conditions for the existence of such subquadratic reductions.

▶ **Theorem 6.4.** *Let D be a set of objects and $D_1, D_2 \subseteq D^n$ be a set of possible sequences. Consider any relation $R \subseteq D \times D$ satisfying the following properties.*

- *There is a data item $\perp$ such that $(\perp, d) \in R$ for all $d \in D$.*
- *There is a data item $\top$ such that $(d, \top) \in R$ for all $d \in D$.*
- *For all $a \in \{1, 2\}$ and any set of data items $(d_1, \dots, d_n) \in D_a$ there is a permutation of indices $i_1, \dots, i_n$ such that for any $j < k$, $(d_{i_j}, d_{i_k}) \notin R$. This ordering can be computed in time $\mathcal{O}(n^{2-\delta})$ for $\delta > 0$. We call this ordering the natural ordering.*

Then Selection$(R^{D_1,D_2}) \leq_2$ ChainLWS(R)*.*

We call a relation satisfying the conditions above a *topological* relation. An immediate corollary is that if we can subquadratically reduce Selection(R) to Selection(R') for some topological relation R', then Selection$(R) \leq_2$ ChainLWS(R').

We conclude by providing interesting instantiations of the subquadratic equivalence of Selection and ChainLWS.

▶ **Corollary 6.5** (NestedBoxes $\equiv_2$ VectorDomination). *The weighted* NestedBoxes *problem on $d = c \log n$ dimensions can be solved in time $n^{2-(1/\mathcal{O}(c \log^2 c))}$. For $d = \omega(logn)$, the (unweighted)* NestedBoxes *problem cannot be solved in time $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$ assuming* **SETH***.*

If we restrict NestedBoxes and VectorDomination to Boolean vectors, then we get SubsetChain and SetContainment, respectively. In this case the upper bound improves to $n^{2-1/\mathcal{O}(\log c)}$ [4]. Note that SetContainment $\equiv_2$ OV, hence SubsetChain $\equiv_2$ OV.

7 Open Problems

We discuss the complexity of some succinct LWS instantiations both from an upper bound and a lower bound perspective by proving equivalences with a number of comparably well-studied core problems. The succinct instantiations we study include natural problems such as LowRankLWS, CC, ChainLWS including NestedBoxes and SubsetChain, as well as previously studied instantiations such as ConcLWS and LIS. A number of open questions remain. Our results do not generalize to arbitrary instantiations of LWS. In particular, Static-LWS does not seem to reduce subquadratically to the problem of finding the minimum element in a succincly descibed matrix. With LowRankLWS and CC we do provide instances for which we can identify equivalent core problems, and it will be interesting to find further examples or even sufficient conditions for which we can reduce LWS to other problems and vice versa.

For the case of ChainLWS, we are able to generalize the reduction from LWS to Selection problems. However, the reduction, while preserving subquadratic algorithms, does not preserve near-linear time algorithms. For some cases, such as LIS, we are able to reconstruct a near-linear time algorithm, which raises the question of what conditions are necessary to do that. Similarly, we give sufficient conditions to reduce from Selection to ChainLWS, and other sufficient or even necessary conditions should be explored for both black-box as well as white-box reductions.

Acknowledgments. We would like to thank Karl Bringmann and Russell Impagliazzo for helpful discussions and comments.

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-time hardness of LCS and other sequence similarity measures. In *Proc. 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS'15)*, pages 59–78, 2015.

2 Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with Edit Distance and friends or: A polylog shaved is a lower bound made. In *Proc. 48th Annual ACM Symposium on Symposium on Theory of Computing (STOC'16)*, 2016. To appear.

3 Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Proc. 41st International Colloquium on Automata, Languages, and Programming (ICALP'14)*, pages 39–51, 2014.

4 Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15)*, pages 218–230, 2015.

5 Alfred V. Aho, Daniel S. Hirschberg, and Jeffrey D. Ullman. Bounds on the complexity of the longest common subsequence problem. *Journal of the ACM*, 23(1):1–12, 1976. doi:10.1145/321921.321922.

6 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proc. 47th Annual ACM Symposium on Theory of Computing (STOC'15)*, pages 51–58, 2015. doi:10.1145/2746539.2746612.

7 David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Patrascu, and Perouz Taslakian. Necklaces, convolutions, and X+Y. *Algorithmica*, 69(2):294–314, 2014. doi:10.1007/s00453-012-9734-3.

8 Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS'14)*, pages 661–670, 2014. doi:10.1109/FOCS.2014.76.

9 Karl Bringmann. A near-linear pseudopolynomial time algorithm for Subset Sum. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*, pages 1073–1084, 2017. doi:10.1137/1.9781611974782.69.

10 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and Dynamic Time Warping. In *Proc. 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS'15)*, pages 79–97, 2015.

11 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the Strong Exponential Time Hypothesis and consequences for non-reducibility. In *Proc. 7th ACM Conference on Innovations in Theoretical Computer Science (ITCS'16)*, pages 261–270, 2016. `doi:10.1145/2840728.2840746`.

12 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In *Proc. 47th Annual ACM Symposium on Theory of Computing, (STOC'15)*, pages 31–40, 2015. `doi:10.1145/2746539.2746568`.

13 Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On problems equivalent to (min,+)-convolution. *ArXiv e-prints*, February 2017. `arXiv:1702.07669`.

14 Mark de Berg, Kevin Buchin, Bart M. P. Jansen, and Gerhard J. Woeginger. Fine-grained complexity analysis of two classic TSP variants. In *Proc. 43rd International Colloquium on Automata, Languages, and Programming (ICALP'16)*, pages 5:1–5:14, 2016. `doi:10.4230/LIPIcs.ICALP.2016.5`.

15 David Eppstein. Sequence comparison with mixed convex and concave costs. *J. Algorithms*, 11(1):85–101, 1990. `doi:10.1016/0196-6774(90)90031-9`.

16 David A. Eppstein. *Efficient algorithms for sequence analysis with concave and convex gap costs.* PhD thesis, Columbia University, 1989.

17 Michael L. Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11(1):29–35, 1975. `doi:10.1016/0012-365X(75)90103-X`.

18 Anka Gajentaan and Mark H Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995.

19 Zvi Galil and Raffaele Giancarlo. Speeding up dynamic programming with applications to molecular biology. *Theoretical Computer Science*, 64(1):107–118, 1989. `doi:10.1016/0304-3975(89)90101-1`.

20 Zvi Galil and Kunsoo Park. A linear-time algorithm for concave one-dimensional dynamic programming. *Inf. Process. Lett.*, 33(6):309–311, 1990. `doi:10.1016/0020-0190(90)90215-J`.

21 Zvi Galil and Kunsoo Park. Parallel algorithms for dynamic programming recurrences with more than $O(1)$ dependency. *J. Parallel Distrib. Comput.*, 21(2):213–222, 1994. `doi:10.1006/jpdc.1994.1053`.

22 Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. Completeness for first-order properties on sparse structures with algorithmic applications. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*, pages 2162–2181, 2017. `doi:10.1137/1.9781611974782.141`.

23 Allan Grønlund, Kasper Green Larsen, Alexander Mathiasen, Jesper Sindahl Nielsen, Stefan Schneider, and Mingzhou Song. Fast Exact k-Means, k-Medians and Bregman Divergence Clustering in 1D. *ArXiv e-prints*, January 2017. `arXiv:1701.07204`.

24 Daniel S. Hirschberg and Lawrence L. Larmore. The least weight subsequence problem. *SIAM Journal on Computing*, 16(4):628–638, 1987. `doi:10.1137/0216043`.

25 Russell Impagliazzo, Shachar Lovett, Ramamohan Paturi, and Stefan Schneider. 0-1 Integer Linear Programming with a linear number of constraints. *ArXiv e-prints*, January 2014. `arXiv:1401.5512`.

26 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

27 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

28 Alon Itai, Christos H Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

29 Maria M. Klawe and Daniel J. Kleitman. An almost linear time algorithm for generalized matrix searching. *SIAM J. Discrete Math.*, 3(1):81–97, 1990. `doi:10.1137/0403009`.

30 Donald E. Knuth and Michael F. Plass. Breaking paragraphs into lines. *Softw., Pract. Exper.*, 11(11):1119–1184, 1981. `doi:10.1002/spe.4380111102`.

31 Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the Fine-grained Complexity of One-Dimensional Dynamic Programming. *ArXiv e-prints*, March 2017. `arXiv:1703.00941`.

32 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

33 William J. Masek and Mike Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20(1):18–31, 1980. `doi:10.1016/0022-0000(80)90002-1`.

34 Jiří Matoušek. Efficient partition trees. *Discrete & Computational Geometry*, 8(1):315–334, 1992.

35 Webb Miller and Eugene W. Myers. Sequence comparison with concave weighting functions. *Bulletin of Mathematical Biology*, 50(2):97–120, 1988. `doi:10.1007/BF02459948`.

36 David Pisinger. Dynamic programming on the word RAM. *Algorithmica*, 35(2):128–145, 2003. `doi:10.1007/s00453-002-0989-y`.

37 Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the Strong Exponential Time Hypothesis (invited talk). In *Proc. 10th International Symposium on Parameterized and Exact Computation (IPEC'15)*, pages 17–29, 2015. `doi:10.4230/LIPIcs.IPEC.2015.17`.

38 Robert E. Wilber. The concave least-weight subsequence problem revisited. *J. Algorithms*, 9(3):418–425, 1988. `doi:10.1016/0196-6774(88)90032-6`.

39 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365, 2005.

40 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *Proc. 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS'10)*, pages 645–654, 2010. `doi:10.1109/FOCS.2010.67`.

41 F. Frances Yao. Efficient dynamic programming using quadrangle inequalities. In *Proc. 12th Annual ACM Symposium on Theory of Computing (STOC'80)*, pages 429–435, 1980. `doi:10.1145/800141.804691`.