

# On Problems Equivalent to $(\min, +)$ -Convolution\*

Marek Cygan<sup>1</sup>, Marcin Mucha<sup>2</sup>, Karol Węgrzycki<sup>3</sup>, and Michał Włodarczyk<sup>4</sup>

1 Institute of Informatics, University of Warsaw, Warsaw, Poland  
cygan@mimuw.edu.pl

2 Institute of Informatics, University of Warsaw, Warsaw, Poland  
mucham@mimuw.edu.pl

3 Institute of Informatics, University of Warsaw, Warsaw, Poland  
k.wegrzycki@mimuw.edu.pl

4 Institute of Informatics, University of Warsaw, Warsaw, Poland  
m.wlodarczyk@mimuw.edu.pl

---

## Abstract

In the recent years, significant progress has been made in explaining apparent hardness of improving over naive solutions for many fundamental polynomially solvable problems. This came in the form of conditional lower bounds – reductions from a problem assumed to be hard. These include 3SUM, All-Pairs Shortest Paths, SAT and Orthogonal Vectors, and others.

In the  $(\min, +)$ -convolution problem, the goal is to compute a sequence  $(c[i])_{i=0}^{n-1}$ , where  $c[k] = \min_{i=0, \dots, k} \{a[i] + b[k - i]\}$ , given sequences  $(a[i])_{i=0}^{n-1}$  and  $(b[i])_{i=0}^{n-1}$ . This can easily be done in  $\mathcal{O}(n^2)$  time, but no  $\mathcal{O}(n^{2-\varepsilon})$  algorithm is known for  $\varepsilon > 0$ . In this paper we undertake a systematic study of the  $(\min, +)$ -convolution problem as a hardness assumption.

As the first step, we establish equivalence of this problem to a group of other problems, including variants of the classic knapsack problem and problems related to subadditive sequences. The  $(\min, +)$ -convolution has been used as a building block in algorithms for many problems, notably problems in stringology. It has also already appeared as an ad hoc hardness assumption. We investigate some of these connections and provide new reductions and other results.

**1998 ACM Subject Classification** F.2 Analysis of Algorithms and Problem Complexity

**Keywords and phrases** fine-grained complexity, knapsack, conditional lower bounds,  $(\min, +)$ -convolution, subquadratic equivalence

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2017.22

## 1 Introduction

### 1.1 Hardness in P

For many problems there exist ingenious algorithms that significantly improve upon the naive approach in terms of time complexity. On the other hand, for some fundamental problems, the naive algorithms are still the best known, or have been improved upon only slightly. To some extent this has been explained by the  $P \neq NP$  conjecture. However, for many problems even the naive approaches lead to polynomial algorithms, and the  $P \neq NP$  conjecture does not seem to be particularly useful for proving polynomial lower bounds.

---

\* This work is part of a project TOTAL that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 677651).



In the recent years, significant progress has been made in establishing such bounds, conditioned on conjectures other than  $P \neq NP$ , each of them claiming time complexity lower bounds for a different problem. And so, conjecture that there is no  $\mathcal{O}(n^{2-\epsilon})$  algorithm for 3SUM problem<sup>1</sup> implies hardness for problems in computational geometry [22] and dynamic algorithms [33]. The conjecture that All-Pairs Shortest Paths (APSP) is hard implies hardness of finding graph radius, graph median and some dynamic problems (see [38] for survey). Finally, the Strong Exponential Time Hypothesis (SETH) introduced in [25, 26] that has been used extensively to prove hardness of parametrized problems, recently lead to polynomial lower bounds via the intermediate Orthogonal Vectors problem (see [36]). These include bounds for Edit Distance [3], Longest Common Subsequence [9, 2], and other [38].

It is worth noting that in many cases the results mentioned are not only showing the hardness of the problem in question, but also that it is computationally equivalent to the underlying hard problem. This leads to clusters of equivalent problems being formed, each cluster corresponding to a single hardness assumption (see [38, Figure 1]).

As Christos H. Papadimitriou is quoted to say „*There is nothing wrong with trying to prove that  $P=NP$  by developing a polynomial-time algorithm for an NP-complete problem. The point is that without an NP-completeness proof we would be trying the same thing without knowing it!*” [32]. In the same spirit, these new conditional hardness results have cleared the polynomial landscape by showing that there really are not that many hard problems.

## 1.2 Hardness of MinConv

In this paper we propose yet another hardness assumption in the MINCONV problem.

MINCONV

**Input:** Sequences  $(a[i])_{i=0}^{n-1}, (b[i])_{i=0}^{n-1}$

**Task:** Output sequence  $(c[i])_{i=0}^{n-1}$ , such that  $c[k] = \min_{i+j=k} (a[i] + b[j])$

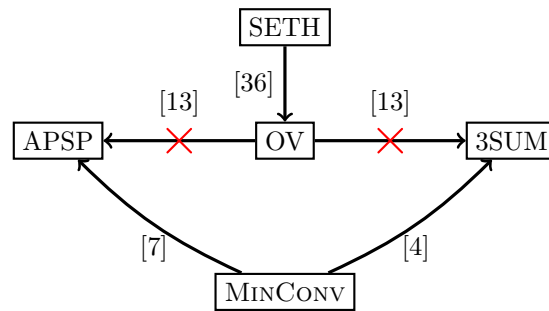
This problem has been used as a hardness assumption before for at least two specific problems [29, 4], but to the best of our knowledge no attempts have been made to systematically study the neighborhood of this problem in the polynomial complexity landscape. To be more precise, we consider the following.

► **Conjecture 1.** *There is no  $\mathcal{O}(n^{2-\epsilon})$  algorithm for MINCONV, for  $\epsilon > 0$ .*

Let us first look at the place occupied by MINCONV in the landscape of established hardness conjectures. Figure 1 shows known reductions between these conjectures and includes MINCONV. Bremner et al. [7] showed reduction from MINCONV to APSP. It is also known [4, 1] that MINCONV can be reduced to 3SUM (to the best of our knowledge no such reduction has been published before, and we provide the details in the full version of this paper [18]). Note that a reduction from 3SUM or APSP to MINCONV would imply a reduction between 3SUM and APSP, which is a major open problem in the area [38]. No relation is known between MINCONV and SETH or OV.

In this paper we study three broad categories of problems. The first category consists of the classic 0/1 KNAPSACK and its variants, which we show to be essentially equivalent to MINCONV. This is perhaps somewhat surprising, given recent progress of Bringmann [8] for SUBSETSUM, which is a special case of 0/1 KNAPSACK. However, note that Bringmann’s

<sup>1</sup> We included all problem definitions together with known results concerning these problems in Section 2. This is to keep the introduction relatively free of technicalities.



■ **Figure 1** The relationship between popular conjectures. A reduction from OV to 3SUM or APSP contradicts the nondeterministic version of SETH [13, 38] (these arrows are striked-out).

algorithm [8] (as well as in other efficient solutions for SUBSETSUM) is built upon the idea of composing solutions using the  $(\vee, \wedge)$ -convolution, which can be implemented efficiently using Fast Fourier Transform (FFT). The corresponding composition operation for 0/1 KNAPSACK is MINCONV (see the full version of this paper for details [18]).

The second category consists of problems directly related to MINCONV. This includes decision versions of MINCONV, and problems related to the notion of subadditivity. Any subadditive sequence  $a$  with  $a[0] = 0$  is an idempotent of MINCONV, so it is perhaps natural that these problems turn out to be equivalent to MINCONV.

Finally, we investigate problems that have previously been shown to be related to MINCONV, and contribute some new reductions, or simplify existing ones.

## 2 Problem definitions and known results

### 2.1 3SUM

3SUM

**Input:** Sets of integers  $A, B, C$ , each of size  $n$

**Task:** Decide whether there exist  $a \in A, b \in B, c \in C$  such that  $a + b = c$

The 3SUM problem is the first problem that was considered as a hardness assumption in P. It admits a simple  $\mathcal{O}(n^2 \log n)$  algorithm but the existence of an  $\mathcal{O}(n^{2-\epsilon})$  algorithm remains a big open problem. The first lower bounds based on hardness of 3SUM appeared in 1995 [22] and some other examples can be found in [5, 33, 39]. The current best algorithm for 3SUM runs in slightly subquadratic expected time  $\mathcal{O}((n^2 / \log^2 n)(\log \log n)^2)$  [5]. An  $\mathcal{O}(n^{1.5} \text{polylog}(n))$  algorithm is possible on the nondeterministic<sup>2</sup> Turing machine [13]. The 3SUM problem is known to be subquadratically equivalent to its convolution version in the randomized setting [33].

3SUMCONV

**Input:** Sequences  $a, b, c$ , each of length  $n$

**Task:** Decide whether there exist  $i, j$  such that  $a[i] + b[j] = c[i + j]$

Both problems are sometimes considered with real weights but in this work we restrict only to the integer setting.

<sup>2</sup> We say that decision problem  $L$  admits a nondeterministic algorithm in time  $T(n)$  if  $L \in \text{NTIME}(T(n)) \cap \text{co-NTIME}(T(n))$ .

## 2.2 MinConv

We have already defined the MINCONV problem in Subsection 1.2. Note that it is equivalent (just by negating elements) to the analogous MAXCONV problem.

MAXCONV

**Input:** Sequences  $(a[i])_{i=0}^{n-1}, (b[i])_{i=0}^{n-1}$

**Task:** Output sequence  $(c[i])_{i=0}^{n-1}$ , such that  $c[k] = \max_{i+j=k}(a[i] + b[j])$

We describe our contribution in terms of MINCONV as this version has been already been heavily studied. However, in the theorems and proofs we use MAXCONV, as it is easier to work with. We will also work with a decision version of the problem.

MAXCONV UPPERBOUND

**Input:** Sequences  $(a[i])_{i=0}^{n-1}, (b[i])_{i=0}^{n-1}, (c[i])_{i=0}^{n-1}$

**Task:** Decide whether  $c[k] \geq \max_{i+j=k}(a[i] + b[j])$  for all  $k$

If we replace the latter condition with  $c[k] \leq \max_{i+j=k}(a[i] + b[j])$  we obtain a similar problem MAXCONV LOWERBOUND. Yet another statement of a decision version asks whether a given sequence is a self upper bound with respect to MAXCONV, i.e., if it is superadditive. From the perspective of MINCONV we may ask an analogous question about being subadditive (again equivalent by negating elements). As far as we know, the computational complexity of these problems has not been studied yet.

SUPERADDITIVITY TESTING

**Input:** A sequence  $(a[i])_{i=0}^{n-1}$

**Task:** Decide whether  $a[k] \geq \max_{i+j=k}(a[i] + a[j])$  for all  $k$

In the standard  $(+, \cdot)$  ring, convolution can be computed in  $\mathcal{O}(n \log n)$  time by the FFT. A natural line of attacking MINCONV would be to design an analogue of FFT in the  $(\min, +)$ -semiring, also called a *tropical semiring*<sup>3</sup>. However, due to the lack of inverse for the min-operation it is unclear if such a transform exists for general sequences. When restricted to convex sequences, one can use a tropical analogue of FFT, namely the Legendre-Fenchel transform [19], which can be performed in linear time [30]. Also, [24] considered sparse variants of convolutions and connection with 3SUM.

There has been a long line of research dedicated to improve  $\mathcal{O}(n^2)$  algorithm for MINCONV. Bremner et al. [7] gave an  $\mathcal{O}(n^2/\log n)$  algorithm for MINCONV, and gave a reduction from MINCONV to APSP [7, Theorem 13]. Williams [37] gave an  $\mathcal{O}(n^3/2^{\Omega(\log n)^{1/2}})$  algorithm for APSP, which implies the best known  $\mathcal{O}(n^2/2^{\Omega(\log n)^{1/2}})$  algorithm for MINCONV [15].

Truly subquadratic algorithms for MINCONV exist for monotone increasing sequences with integer values bounded by  $\mathcal{O}(n)$ . Chan and Lewenstein [15] presented an  $\mathcal{O}(n^{1.859})$  randomized algorithm and an  $\mathcal{O}(n^{1.864})$  deterministic algorithm for that case. They exploited ideas from additive combinatorics. Bussieck et al. [12] showed that for random input, MINCONV can be computed in  $\mathcal{O}(n \log n)$  expected and  $\Theta(n^2)$  worst case time.

If we are satisfied with computing  $c$  with a relative error  $(1 + \epsilon)$  then general MINCONV admits a nearly-linear algorithm [4, 40]. It could be called an FPTAS (fully polynomial-time

<sup>3</sup> In this setting MINCONV is often called  $(\min, +)$ -convolution, inf-convolution or epigraphic sum.

approximation schema) with a remark that usually this name is reserved for single-output problems for which decision versions are NP-hard.

Using techniques of Carmosino et al. [13] and reduction from MAXCONV UPPERBOUND to 3SUM one can construct an  $\mathcal{O}(n^{1.5} \text{polylog}(n))$  algorithm working on nondeterministic Turing machines for MAXCONV UPPERBOUND (see the full version of this paper [18]). This running time matches the  $\mathcal{O}(n^{1.5})$  algorithm for MINCONV in the nonuniform decision tree model [7]. This result is based on the techniques of Fredman [21, 20]. It remains unclear how to transfer these results to the word-RAM model [7].

## 2.3 Knapsack

### 0/1 KNAPSACK

**Input:** A set of items  $\mathcal{I}$  with given weights and values  $((w_i, v_i))_{i \in \mathcal{I}}$ , capacity  $t$

**Task:** Find the maximal total value of the items subset  $\mathcal{I}' \subseteq \mathcal{I}$  such that  $\sum_{i \in \mathcal{I}'} w_i \leq t$

If we are allowed to take multiple copies of a single item then we obtain the UNBOUNDED KNAPSACK problem. The decision versions of both problems are known to be NP-hard [23] but there are classical algorithms based on dynamic programming with a pseudo-polynomial running time  $\mathcal{O}(nt)$  [6]. In fact they solve more general problems, i.e., 0/1 KNAPSACK<sup>+</sup> and UNBOUNDED KNAPSACK<sup>+</sup>, where we are asked to output answers for each  $0 < t' \leq t$ . There is also a long line of research on FPTAS for KNAPSACK with the current best running times respectively  $\mathcal{O}(n \log \frac{1}{\epsilon} + \frac{1}{\epsilon^3} \log^2 \frac{1}{\epsilon})$  for 0/1 KNAPSACK [28] and  $\mathcal{O}(n + \frac{1}{\epsilon^2} \log^3 \frac{1}{\epsilon})$  for UNBOUNDED KNAPSACK [27].

## 2.4 Other problems related to MinConv

### TREE SPARSITY

**Input:** A rooted tree  $T$  with a weight function  $x : V(T) \rightarrow \mathbb{N}_{\geq 0}$ , parameter  $k$

**Task:** Find the maximal total weight of rooted subtree of size  $k$

The TREE SPARSITY problem admits an  $\mathcal{O}(nk)$  algorithm, which was at first invented for restricted case of balanced trees [14] and generalised later [4]. There is also a nearly-linear FPTAS based on the FPTAS for MINCONV [4]. It is known that an  $\mathcal{O}(n^{2-\epsilon})$  algorithm for TREE SPARSITY entails a subquadratic algorithm for MINCONV [4].

### MCSP

**Input:** A sequence  $(a[i])_{i=0}^{n-1}$

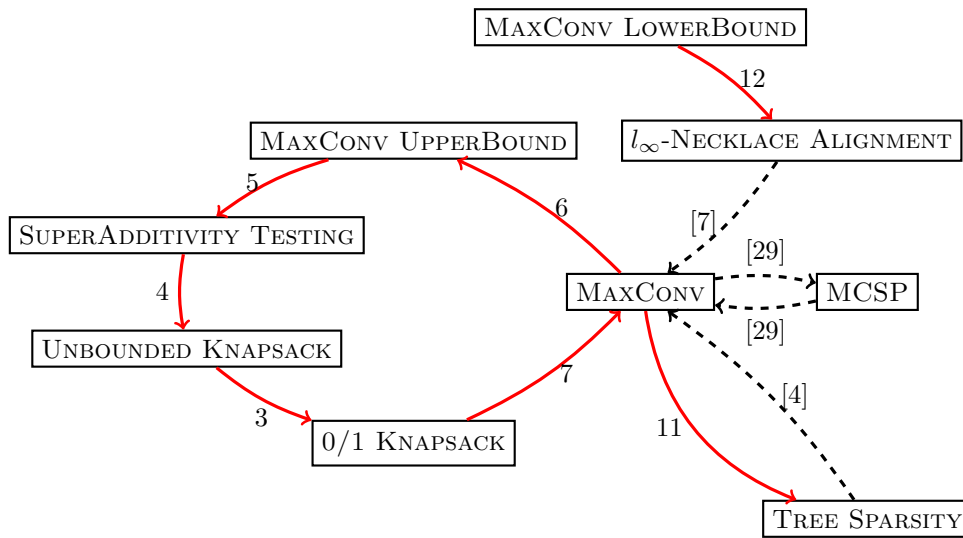
**Task:** Output the maximal sum of  $k$  consecutive elements for each  $k$

There is a trivial  $\mathcal{O}(n^2)$  algorithm for MCSP and a nearly-linear FPTAS based on the FPTAS for MINCONV [16]. To the best of our knowledge, this is the first problem to have been explicitly proven to be subquadratically equivalent with MINCONV [29]. Our reduction to SUPERADDITIVITY TESTING allows us to significantly simplify the proof (see Section 6.1).

### $l_p$ -NECKLACE ALIGNMENT

**Input:** Sequences  $(x[i])_{i=0}^{n-1}, (y[i])_{i=0}^{n-1}$  describing locations of beads on a circle

**Task:** Output the cost of the best alignment in  $p$ -norm, i.e.,  $\sum_{i=0}^{n-1} d(x[i] + c, y[\pi(i)])^p$  where  $c$  is a circular shift,  $\pi$  is a permutation, and  $d$  is a distance function on a circle



■ **Figure 2** Summary of reductions in the MINCONV complexity class. An arrow from problem  $A$  to  $B$  denotes a reduction from  $A$  to  $B$ . Black dashed arrows were previously known, red arrows are new results. Numbers next to red arrows point to the corresponding theorems. The only randomized reduction is in the proof of Theorem 7.

For  $p = \infty$  we are interested in bounding the maximal distance between any two matched beads. The problem initially emerged for  $p = 1$  during the research on geometry of musical rhythm [35]. The family of NECKLACE ALIGNMENT problems has been systematically studied by Bremner et al. [7] for various values of  $p$ , in particular 1, 2,  $\infty$ . For  $p = 2$  they presented an  $\mathcal{O}(n \log n)$  algorithm based on Fast Fourier Transform. For  $p = \infty$  the problem was reduced to MINCONV which led to a slightly subquadratic algorithm.

Although it is more natural to state the problem with inputs from  $[0, 1)$ , we find it more convenient to work with integer sequences that describe a necklace after scaling.

Fast  $o(n^2)$  algorithms for MINCONV have also found applications in text algorithms. Moosa and Rahman [31] reduced the *Indexed Permutation Matching* to MINCONV and obtained  $o(n^2)$  algorithm. Burcsi et al. [10] used MINCONV to get faster algorithms for *Jumbled Pattern Matching* and described how finding dominating pairs can be used to solve MINCONV. Later Burcsi et al. [11] showed that fast MINCONV can also be used to get faster algorithms for a decision version of the *Approximate Jumbled Pattern Matching* over binary alphabets.

### 3 New results summary

Figure 2 illustrates the technical contributions of this paper. The long ring of reductions on the left side of the Figure 2 is summarized below.

► **Theorem 2.** *The following statements are equivalent:*

1. *There exists an  $\mathcal{O}(n^{2-\varepsilon})$  algorithm for MAXCONV for some  $\varepsilon > 0$ .*
2. *There exists an  $\mathcal{O}(n^{2-\varepsilon})$  algorithm for MAXCONV UPPERBOUND for some  $\varepsilon > 0$ .*
3. *There exists an  $\mathcal{O}(n^{2-\varepsilon})$  algorithm for SUPERADDITIVITY TESTING for some  $\varepsilon > 0$ .*
4. *There exists an  $\mathcal{O}((n+t)^{2-\varepsilon})$  algorithm for UNBOUNDED KNAPSACK for some  $\varepsilon > 0$ .*
5. *There exists an  $\mathcal{O}((n+t)^{2-\varepsilon})$  algorithm for 0/1 KNAPSACK for some  $\varepsilon > 0$ .*

*We allow randomized algorithms.*

Theorem 2 is split into five implications, presented separately as Theorems 3,4,5,6 and 7 in Section 5. While Theorem 2 has a relatively short and simple statement, it is not the strongest possible version of the equivalence. In particular, one can show analogous implications for subpolynomial improvements, such as the  $\mathcal{O}(n^2/2^{\Omega(\log n)^{1/2}})$  algorithm for MINCONV of Williams [37]. The theorems listed above contain stronger versions of the implications.

Section 6 is devoted to the remaining arrows in Figure 2. In Subsection 6.1, we show that by using Theorem 2 we can obtain an alternative proof of the equivalence of MCSP and MAXCONV (and so also MINCONV), much simpler than the one presented in [29]. In Subsection 6.2, we show that TREE SPARSITY reduces to MAXCONV, complementing the opposite reduction showed in [4]. Finally in Subsection 6.3 we provide some observations on the possible equivalence between  $l_\infty$ -NECKLACE ALIGNMENT and MAXCONV.

## 4 Preliminaries

We present a series of results of the following form: if a problem  $\mathcal{A}$  admits an algorithm with running time  $T(n)$ , then a problem  $\mathcal{B}$  admits an algorithm with running time  $T'(n)$ , where function  $T'$  depends on  $T$  and  $n$  is the length of the input. Our main interest is in showing that  $T(n) = \mathcal{O}(n^{2-\epsilon}) \Rightarrow T'(n) = \mathcal{O}(n^{2-\epsilon'})$ . Some problems, in particular KNAPSACK, have no simple parameterization and we allow function  $T$  to take multiple arguments.

We assume that for all studied problems the input consists of a list of integers within  $[-W, W]$ . For the sake of readability we omit  $W$  as a running time parameter and we allow function  $T$  to hide polylog( $W$ ) factors. As sometimes the size of the input grows in the reduction, we restrict ourselves to a class of functions satisfying  $T(cn) = \mathcal{O}(T(n))$  for a constant  $c$ . This is justified as we mainly focus on functions of the form  $T(n) = n^\alpha$ . In some reductions the integers in the new instance may increase to  $\mathcal{O}(nW)$ . In that case we multiply the running time by polylog( $n$ ) to take into account the overhead of performing arithmetic operations. All logarithms are base 2.

## 5 Main reductions

► **Theorem 3** (UNBOUNDED KNAPSACK  $\rightarrow$  0/1 KNAPSACK). *A  $T(n, t)$  algorithm for 0/1 KNAPSACK implies an  $\mathcal{O}(T(n, t) \log t)$  algorithm for UNBOUNDED KNAPSACK.*

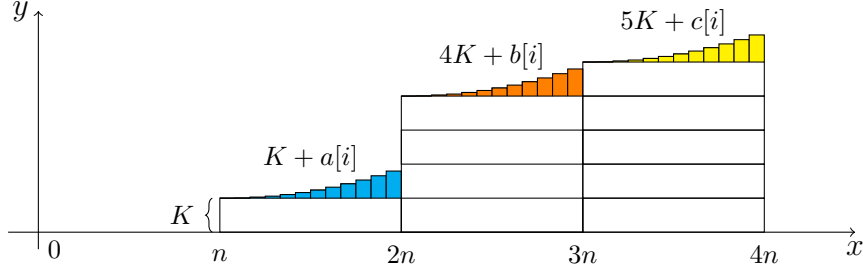
**Proof.** Consider an instance of UNBOUNDED KNAPSACK with the capacity  $t$  and the set of items given as weight-value pairs  $((w_i, v_i))_{i \in \mathcal{I}}$ . Construct an equivalent 0/1 KNAPSACK instance with the same  $t$  and the set of items  $((2^j w_i, 2^j v_i))_{i \in \mathcal{I}, 0 \leq j \leq \log t}$ . Let  $X = (x_i)_{i \in \mathcal{I}}$  be the list of multiplicities of items chosen in a solution to the UNBOUNDED KNAPSACK problem. Of course  $x_i \leq t$ . Define  $(x_i^j)_{0 \leq j \leq \log t}$ ,  $x_i^j \in \{0, 1\}$  to be the binary representation of  $x_i$ . Then the vector  $(x_i^j)_{i \in \mathcal{I}, 0 \leq j \leq \log t}$  induces a solution to 0/1 KNAPSACK with the same total weight and value. The described mapping can be reverted what implies the equivalence between the instances and proves the claim. ◀

► **Theorem 4** (SUPERADDITIVITY TESTING  $\rightarrow$  UNBOUNDED KNAPSACK). *If UNBOUNDED KNAPSACK can be solved in time  $T(n, t)$  then SUPERADDITIVITY TESTING admits an algorithm with running time  $\mathcal{O}(T(n, n) \log n)$ .*

**Proof.** Let  $(a[i])_{i=0}^{n-1}$  be a non-negative monotonic sequence.<sup>4</sup> Set  $D = \sum_{i=0}^{n-1} a[i] + 1$  and construct an UNBOUNDED KNAPSACK instance with the set of items  $((i, a[i]))_{i=0}^{n-1} \cup$

<sup>4</sup> For a technical reduction of SUPERADDITIVITY TESTING to this case see the full version of this paper [18].





■ **Figure 3** Graphical interpretation of the sequence  $e$  in Theorem 5. The height of rectangles equals  $K$ .

$((2n - 1 - i, D - a[i]))_{i=0}^{n-1}$  and  $t = 2n - 1$ . It is always possible to gain  $D$  by taking two items  $(i, a[i]), (2n - 1 - i, D - a[i])$  for any  $i$ . We will claim that the answer to the constructed instance equals  $D$  if and only if  $a$  is superadditive.

If  $a$  is not superadditive, then there are  $i, j$  such that  $a[i] + a[j] > a[i + j]$ . Choosing  $((i, a[i]), (j, a[j]), (2n - 1 - i - j, D - a[i + j]))$  gives a solution of value exceeding  $D$ .

Now assume that  $a$  is superadditive. Observe that any feasible knapsack solution may contain at most one item with weight exceeding  $n - 1$ . On the other hand, the optimal solution has to include one such item because the total value of the lighter ones is less than  $D$ . Therefore the optimal solution contains an item  $(2n - 1 - k, D - a[k])$  for some  $k < n$ . The total weight of the rest of the solution is at most  $k$ . As  $a$  is superadditive, we can replace any pair  $(i, a[i]), (j, a[j])$  with the item  $(i + j, a[i + j])$  without decreasing the value of the solution. By repeating this argument, we end up with a single item lighter than  $n$ . The sequence  $a$  is monotonic so it is always profitable to replace this item with a heavier one, as long as the load does not exceed  $t$ . We conclude that the optimal solution must be of form  $((k, a[k]), (2n - 1 - k, D - a[k]))$ , which completes the proof. ◀

► **Theorem 5** (MAXCONV UPPERBOUND  $\rightarrow$  SUPERADDITIVITY TESTING). *If SUPERADDITIVITY TESTING can be solved in time  $T(n)$  then MAXCONV UPPERBOUND admits an algorithm with running time  $\mathcal{O}(T(n) \log n)$ .*

**Proof.** We start with reducing the instance of MAXCONV UPPERBOUND to the case of non-negative monotonic sequences. Observe that condition  $a[i] + b[j] \leq c[i + j]$  can be rewritten as  $(C + a[i] + Di) + (C + b[j] + Dj) \leq 2C + c[i + j] + D(i + j)$  for any constants  $C, D$ . Hence, replacing sequences  $(a[i])_{i=0}^{n-1}, (b[i])_{i=0}^{n-1}, (c[i])_{i=0}^{n-1}$  with  $a'[i] = C + a[i] + Di, b'[i] = C + b[i] + Di, c'[i] = 2C + c[i] + Di$  leads to an equivalent instance. We can thus pick  $C, D$  of magnitude  $\mathcal{O}(W)$  to ensure that all elements are non-negative and do not exceed the successor. The values in the new sequences may rise up to  $\mathcal{O}(nW)$ .

From now we can assume the given sequences to be non-negative and monotonic. Define  $K$  to be the maximal value occurring in any sequence. Construct a sequence  $e$  of length  $4n$  as follows. For  $i \in [0, n - 1]$  set  $e[i] = 0, e[n + i] = K + a[i], e[2n + i] = 4K + b[i], e[3n + i] = 5K + c[i]$ . If there is  $a[i] + b[j] > c[i + j]$  for some  $i, j$ , then  $e[n + i] + e[2n + j] > e[3n + i + j]$  and therefore  $e$  is not superadditive. We now show that otherwise  $e$  must be superadditive.

Assume w.l.o.g.  $i \leq j$ . The case  $i < n$  can be ruled out because it implies  $e[i] = 0$  and  $e[i] + e[j] \leq e[i + j]$  for any  $j$  as  $e$  is monotonic. If  $i \geq 2n$ , then  $i + j \geq 4n$ , so we can restrict to  $i \in [n, 2n - 1]$ . We can also clearly assume  $j < 3n$ . If  $j \in [n, 2n - 1]$ , then  $e[i] + e[j] \leq 4K \leq e[i + j]$ . Finally,  $j \in [2n, 3n - 1]$  corresponds to the original condition. ◀



► **Theorem 6** (MAXCONV  $\rightarrow$  MAXCONV UPPERBOUND). *A  $T(n)$  algorithm for MAXCONV UPPERBOUND implies an  $\mathcal{O}(T(\sqrt{n})n \log n)$  algorithm for MAXCONV.*

The proof of the reduction from MAXCONV to MAXCONV UPPERBOUND has been independently given recently in [4]. For completeness we give our proof in the full version of this paper [18].

► **Theorem 7** (0/1 KNAPSACK  $\rightarrow$  MAXCONV). *A  $T(n)$  algorithm for MAXCONV implies an  $\mathcal{O}(T(t \log t) \log^3(n/\delta) \log n)$  for 0/1 KNAPSACK that outputs the correct answer with probability at least  $1 - \delta$ .*

► **Corollary 8.** *The  $\mathcal{O}((n+t)^{2-\epsilon})$  time algorithm for 0/1 KNAPSACK implies the randomized  $\mathcal{O}(t^{2-\epsilon'} + n)$  time algorithm for 0/1 KNAPSACK<sup>+</sup>.*

The proof follows the approach of Bringmann [8], and we present it in the full version of this paper [18].

## 6 Other problems related to MinConv

### 6.1 Maximum consecutive subsums problem

The MAXIMUM CONSECUTIVE SUBSUMS PROBLEM (MCSP) is to the best of our knowledge the first problem that has been explicitly proven to be subquadratically equivalent with MINCONV [29]. The reduction from MCSP to MAXCONV is only shown for completeness, but the reduction in the opposite direction is much simpler than the original one.

► **Theorem 9** (MCSP  $\rightarrow$  MAXCONV). *If MAXCONV can be solved in time  $T(n)$  then MCSP admits an algorithm with running time  $\mathcal{O}(T(n))$ .*

**Proof.** Let  $(a[i])_{i=0}^{n-1}$  be the input sequence. Construct sequences of length  $2n$  as follows:  $b[k] = \sum_{i=0}^k a[i]$  for  $k < n$ ,  $c[k] = -\sum_{i=0}^{n-k-1} a[i]$  for  $k \leq n$  (empty sum equals 0) and otherwise  $b[k] = c[k] = -D$ , where  $D$  is two times larger than any partial sum. Observe that

$$(b \oplus^{\max} c)[n+k-1] = \max_{\substack{0 \leq j < n \\ 0 \leq n+k-j-1 \leq n}} \sum_{i=0}^j a[i] - \sum_{i=0}^{j-k} a[i] = \max_{k-1 \leq j < n} \sum_{i=j-k+1}^j a[i], \quad (1)$$

so we can read the maximum consecutive sum for each length  $k$  after performing MAXCONV. ◀

► **Theorem 10** (SUPERADDITIVITY TESTING  $\rightarrow$  MCSP). *If MCSP can be solved in time  $T(n)$  then SUPERADDITIVITY TESTING admits an algorithm with running time  $\mathcal{O}(T(n))$ .*

**Proof.** Let  $(a[i])_{i=0}^{n-1}$  be the input sequence and  $b[i] = a[i+1] - a[i]$ . The superadditivity condition  $a[k] \leq a[k+j] - a[j]$  (for all possible  $k, j$ ) can be translated into  $a[k] \leq \min_{0 \leq j < n-k} \sum_{i=j}^{k+j-1} b[i]$  (for all  $k$ ), so computing MCSP vector on  $(-b[i])_{i=0}^{n-2}$  suffices to check if the above condition holds. ◀

### 6.2 Tree Sparsity

► **Theorem 11** (TREE SPARSITY  $\rightarrow$  MAXCONV). *If MAXCONV can be solved in time  $T(n)$  and the function  $T$  is superadditive then TREE SPARSITY admits an algorithm with running time  $\mathcal{O}(T(n) \log^2 n)$ .*

**Proof.** We take advantage of the heavy-light decomposition introduced by Sleator and Tarjan [34]. This technique has been utilized by Backurs et al. [4] in order to transform a nearly-linear PTAS for MAXCONV to a nearly-linear PTAS for TREE SPARSITY. The reduction for exact subquadratic algorithms is different in the second phase though.

We construct a *spine* with a *head*  $s_1$  at the root of the tree. We define  $s_{i+1}$  to be the child of  $s_i$  with the larger subtree (in case of draw we choose any child) and the last node in the spine is a leaf. The remaining children of nodes  $s_i$  become heads for analogous spines so the whole tree gets covered. Note that every path from a leaf to the root intersects at most  $\log n$  spines because each spine transition doubles the subtree size.

For a node  $v$  with a subtree of size  $m$  we define the sparsity vector  $(x^v[0], x^v[1], \dots, x^v[m])$  with the weights of the heaviest subtrees rooted at  $v$  with fixed sizes. We are going to compute sparsity vectors for all heads of spines in the tree recursively. Let  $(s_i)_{i=1}^\ell$  be a spine with a head  $v$  and let  $u^i$  indicate the sparsity vector for the child of  $s_i$  being a head (i.e., the child with the smaller subtree). If  $s_i$  has less than two children we treat  $u^i$  as a vector  $(0)$ .

For an interval  $[a, b] \subseteq [1, \ell]$  let  $u^{a,b} = u^a \oplus^{\max} u^{a+1} \oplus^{\max} \dots \oplus^{\max} u^b$  and  $y^{a,b}[k]$  be the maximum weight of a subtree of size  $k$  rooted at  $s_a$  and not containing  $s_{b+1}$ . Let  $c = \lfloor \frac{a+b}{2} \rfloor$ . The  $\oplus^{\max}$  operator is associative so  $u^{a,b} = u^{a,c} \oplus^{\max} u^{c+1,b}$ . To compute the second vector we consider two cases: whether the optimal subtree contains  $s_{c+1}$  or not.

$$\begin{aligned} y^{a,b}[k] &= \max \left[ y^{a,c}[k], \sum_{i=a}^c x(s_i) + \max_{k_1+k_2=k-(c-a+1)} \left( u^{a,c}[k_1] + y^{c+1,b}[k_2] \right) \right] \\ &= \max \left[ y^{a,c}[k], \sum_{i=a}^c x(s_i) + \left( u^{a,c} \oplus^{\max} y^{c+1,b} \right) [k - (c - a + 1)] \right] \end{aligned}$$

Using the presented formulas we reduce the problem of computing  $x^v = y^{1,\ell}$  to subproblems for intervals  $[1, \frac{\ell}{2}]$  and  $[\frac{\ell}{2} + 1, \ell]$  and results are merged with two  $(\max, +)$ -convolutions. Proceeding further we obtain  $\log \ell$  levels of recursion, where the sum of convolution sizes on each level is  $\mathcal{O}(m)$ , what results in the total running time  $\mathcal{O}(T(m) \log m)$  (recall that  $T$  is superadditive).

The second type of recursion comes from the spine decomposition. There are at most  $\log n$  levels of recursion with the cumulative sum of subtrees bounded by  $n$  on each level, what proves the claim.  $\blacktriangleleft$

### 6.3 $l_\infty$ -Necklace Alignment

In this section we study the  $l_\infty$ -NECKLACE ALIGNMENT alignment problem that was proved to reduce to MINCONV [7]. We are unable to reduce any of the problems equivalent to MINCONV to this problem, but we do reduce a related problem - MAXCONV LOWERBOUND. We also elaborate on why obtaining a full reduction is difficult.

**► Theorem 12** (MAXCONV LOWERBOUND  $\rightarrow$   $l_\infty$ -NECKLACE ALIGNMENT). *If  $l_\infty$ -NECKLACE ALIGNMENT can be solved in time  $T(n)$  then MAXCONV LOWERBOUND admits an algorithm with running time  $\mathcal{O}(T(n) \log n)$ .*

**Proof.** Let  $a, b, c$  be the input sequences to MAXCONV LOWERBOUND. We call a sum of form  $e_1[k_1] + e_2[k_2] + \dots + e_m[k_m]$ , where  $e_i \in \{a, b, c\}$ , a *combination*, and we define its order as  $\sum_{i=1}^m k_i$ . If an element  $e_i[k_i]$  occurs with minus, we subtract  $k_i$ .

We can assume the following properties of the input sequences w.l.o.g.

1. We may assume the sequences are non-negative and  $a[i] \leq c[i]$  for all  $i$ . Just add  $C_1$  to  $a$ ,  $C_1 + C_2$  to  $b$ , and  $2C_1 + C_2$  to  $c$  for appropriate positive constants  $C_1, C_2$ .

2. We can artificially append an element  $b[n]$  larger than value of any combination of order  $n$  and length bounded by a constant  $L$ . Alternatively, we can say that combinations of order 0 with a positive coefficient at  $b[n]$  have positive value. Initially, we enforce this property by setting  $b[n]$  as the maximum absolute value of an element times  $L$ .
3. Any combination of positive order and length bounded by  $L$  has a non-negative value. Add a linear function  $Di$  to all sequences. As the order of combination is positive, the factors at  $D$  sum up to a positive value. It suffices to choose  $D$  equal to the maximum absolute value of an element times  $L$ . Note that previous inequalities compare combinations of the same order so they stay unaffected.

The values of the elements might increase to  $\mathcal{O}(nWL^2)$ . For the rest of this proof we will use  $L = 10$ . Let  $B = b[n]$ ,  $B_1 = b[n - 1]$ ,  $B_2 = b[n] - b[1]$ . We define necklaces  $x, y$  of length  $2B$  with  $2n$  beads each. The property (3) implies monotonicity of the sequences so the beads are given in the right order. We allow two beads to lie in the same place (in particular the first one and the last one in  $y$ ).

$$\begin{aligned} x &= ( a[0], & a[1], & \dots, & a[n-1], & B+c[0], & B+c[1], & \dots, & B+c[n-2], & B+c[n-1] ), \\ y &= ( B_1-b[n-1], & B_1-b[n-2], & \dots, & B_1-b[0], & B+B_2-b[n-1], & B+B_2-b[n-2], & \dots, & B+B_2-b[1], & 2B ). \end{aligned}$$

Bremner et al. [7] pointed out that the optimal solution for  $l_\infty$ -NECKLACE ALIGNMENT must be non-crossing, so we can consider only matchings of form  $(x[i], y[j])$  where  $j = i + k \bmod 2n$  and  $k$  is fixed. Let  $d(x[i], y[j])$  be the forward distance between  $x[i]$  and  $y[j]$ , i.e.,  $y[j] - x[i]$  plus the length of the necklaces if  $j < i$ . Define  $M_k$  to be  $\max_{i \in [0, N]} d(x[i], y[k + i \bmod 2n]) - \min_{i \in [0, N]} d(x[i], y[k + i \bmod 2n])$ . In this setting [7, Fact 5] says that for a fixed  $k$  the optimal shift provides solution of value  $\frac{M_k}{2}$ .

We want to show that for  $k \in [0, n)$  it holds

$$\begin{aligned} \min_{i \in [0, 2n)} d(x[i], y[k + i \bmod 2n]) &= B_1 - \max_{i+j=n-k-1} (a[i] + b[j]), \\ \max_{i \in [0, 2n)} d(x[i], y[k + i \bmod 2n]) &= B - c[n - k - 1]. \end{aligned}$$

There are five types of connections between beads.

$$d(x[i], y[k + i \bmod 2n]) = \begin{cases} B_1 - a[i] - b[n - k - 1 - i] & i \in [0, n - k - 1], & \text{(I)} \\ B + B_2 - a[i] - b[2n - k - 1 - i] & i \in [n - k, n - 1], & \text{(II)} \\ B_2 - b[2n - k - 1 - i] - c[i - n] & i \in [n, 2n - k - 2], & \text{(III)} \\ B - c[n - k - 1] & i = 2n - k - 1, & \text{(IV)} \\ B + B_1 - b[3n - k - 1 - i] - c[i - n] & i \in [2n - k, 2n - 1]. & \text{(V)} \end{cases}$$

All formulas form combinations of length bounded by 5 so we can apply the properties (2,3). Observe that the order of each combination equals  $k$ , except for  $i = 2n - k - 1$  where the order is  $k + 1$ . Using the property (3) we reason that  $B - c[n - k - 1]$  is indeed the maximal forward distance. It remains to show that the minimum lies within the group (I). Note that these are the only combinations that lack  $b[n]$ . By the property (2) each distance from the group (I) compares less with any other distance because the combinations have the same order (except for the maximal one) and only the latter contains  $b[n]$ .

For  $k < n$  the condition  $M_k < B - B_1$  is equivalent to  $c[n - k - 1] > \max_{i+j=n-k-1} (a[i] + b[j])$ . If there is such a  $k$ , i.e., the answer to MAXCONV LOWERBOUND for sequences  $a, b, c$  is NO, then  $\min_k M_k < B - B_1$  and the return value is less than  $\frac{1}{2}(B - B_1)$ .

Finally, we need to prove that otherwise  $M_k \geq B - B_1$  for all  $k$ . We have already acknowledged that for  $k < n$ . Each matching for  $k \geq n$  can be represented as swapping sequences  $a$  and  $c$  inside the necklace  $x$ , composed with the index shift by  $k - n$ . The two halves of the necklace  $x$  are analogous so all the prior observations on the matching structure remain valid.

If the answer to MAXCONV LOWERBOUND for sequences  $a, b, c$  is YES, then  $\forall_{k \in [0, n]} \exists_{i+j=k} a[i] + b[j] \geq c[k]$ . The property (1) guarantees that  $a \leq c$  so we conclude that  $\forall_{k \in [0, n]} \exists_{i+j=k} c[i] + b[j] \geq a[i] + b[j] \geq c[k] \geq a[k]$ , and by the same argument as before the cost of the solution is at least  $\frac{1}{2}(B - B_1)$ .  $\blacktriangleleft$

Observe that both  $l_\infty$ -NECKLACE ALIGNMENT and MAXCONV LOWERBOUND admit simple linear nondeterministic algorithms. For MAXCONV LOWERBOUND it is enough to either assign each  $k$  a single condition  $a[i] + b[k - i] \geq c[k]$  that is satisfied, or guess a  $k$  for which none inequality holds. For  $l_\infty$ -NECKLACE ALIGNMENT we define a decision version of the problem by asking if there is an alignment of value bounded by  $K$  (the problem is self-reducible via binary search). For positive instances the algorithm just guesses  $k$  inducing an optimal solution. For negative instances it must hold  $M_k > 2K$  for all  $k$ . Therefore, it suffices to guess for each  $k$  a pair  $i, j$  such that  $d(x[i], y[k + i \bmod n]) - d(x[j], y[k + j \bmod n]) > 2K$ .

We remind that MAXCONV UPPERBOUND reduces to 3SUM which admits an  $\mathcal{O}(n^{1.5} \text{polylog}(n))$  nondeterministic algorithm [13] so in fact there is no obstacle for a subquadratic reduction from MAXCONV LOWERBOUND to MAXCONV UPPERBOUND to exist (see the full version of this paper [18]). However, the nondeterministic algorithm for 3SUM exploits techniques significantly different from ours, including modular arithmetic, and a potential reduction would probably need to rely on some different structural properties of MAXCONV.

## 7 Conclusions and future work

In this paper we undertake a systematic study of MINCONV as a hardness assumption, and prove subquadratic equivalence of MINCONV with SUPERADDITIVITY TESTING, UNBOUNDED KNAPSACK, 0/1 KNAPSACK, and TREE SPARSITY. An intriguing open problem is to establish the relation between the MINCONV conjecture and SETH.

One consequence of our results is a new lower bound on 0/1 KNAPSACK. It is known that an  $\mathcal{O}(t^{1-\epsilon} n^{\mathcal{O}(1)})$  algorithm for 0/1 KNAPSACK contradicts the SETCOVER conjecture [17]. Here, we show that an  $\mathcal{O}((n+t)^{2-\epsilon})$  algorithm contradicts the MINCONV conjecture. This does not rule out an  $\mathcal{O}(t + n^{\mathcal{O}(1)})$  algorithm, which leads to another interesting open problem.

Finally, it is open whether MAXCONV LOWERBOUND is equivalent to MINCONV, which would imply an equivalence between  $l_\infty$ -NECKLACE ALIGNMENT and MINCONV.

**Acknowledgements.** We would like to thank Amir Abboud, Karl Bringmann and Virginia Vassilevska Williams for helpful discussions.

---

## References

- 1 Amir Abboud. Personal communication.
- 2 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 59–78. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.14.
- 3 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58. ACM, 2015. doi:10.1145/2746539.2746612.

- 4 Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. Better approximations for tree sparsity in nearly-linear time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2215–2229. SIAM, 2017.
- 5 Ilya Baran, Erik D. Demaine, and Mihai Pătraşcu. Subquadratic Algorithms for 3SUM. In *Proceedings of the 9th International Conference on Algorithms and Data Structures, WADS'05*, pages 409–421, Berlin, Heidelberg, 2005. Springer-Verlag. doi:10.1007/11534273\_36.
- 6 Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1957.
- 7 David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, and Perouz Taslakian. Necklaces, Convolutions, and  $X + Y$ . In Yossi Azar and Thomas Erlebach, editors, *Algorithms – ESA 2006: 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006. Proceedings*, pages 160–171, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- 8 Karl Bringmann. A near-linear pseudopolynomial time algorithm for subset sum. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1073–1084. SIAM, 2017.
- 9 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 79–97. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.15.
- 10 Peter Burcsi, Ferdinando Cicalese, Gabriele Fici, and Zsuzsanna Lipták. On table arrangements, scrabble freaks, and jumbled pattern matching. In Paolo Boldi and Luisa Gargano, editors, *Fun with Algorithms, 5th International Conference, FUN 2010, Ischia, Italy, June 2-4, 2010. Proceedings*, volume 6099 of *Lecture Notes in Computer Science*, pages 89–101. Springer, 2010. doi:10.1007/978-3-642-13122-6\_11.
- 11 Peter Burcsi, Ferdinando Cicalese, Gabriele Fici, and Zsuzsanna Lipták. On approximate jumbled pattern matching in strings. *Theory Comput. Syst.*, 50(1):35–51, 2012. doi:10.1007/s00224-011-9344-5.
- 12 Michael R. Bussieck, Hannes Hassler, Gerhard J. Woeginger, and Uwe T. Zimmermann. Fast algorithms for the maximum convolution problem. *Oper. Res. Lett.*, 15(3):133–141, 1994. doi:10.1016/0167-6377(94)90048-5.
- 13 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 Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 261–270. ACM, 2016. doi:10.1145/2840728.2840746.
- 14 Coralia Cartis and Andrew Thompson. An exact tree projection algorithm for wavelets. *IEEE Signal Processing Letters*, 20(11):1026–1029, 2013.
- 15 Timothy M. Chan and Moshe Lewenstein. Clustered Integer 3SUM via Additive Combinatorics. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing, STOC'15*, pages 31–40, New York, NY, USA, 2015. ACM. doi:10.1145/2746539.2746568.
- 16 Ferdinando Cicalese, Eduardo Sany Laber, Oren Weimann, and Raphael Yuster. Approximating the maximum consecutive subsums of a sequence. *Theor. Comput. Sci.*, 525:130–137, 2014. doi:10.1016/j.tcs.2013.05.032.
- 17 Marek Cygan, Holger Dell, Daniel Lokshtanov, Dañiel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlstrom. On problems as hard as cnf-sat. In *Proceedings of the 2012 IEEE Conference on Computational Complex-*

- ity (CCC)*, CCC'12, pages 74–84, Washington, DC, USA, 2012. IEEE Computer Society. doi:10.1109/CCC.2012.36.
- 18 Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Włodarczyk. On problems equivalent to  $(\min,+)$ -convolution. *CoRR*, abs/1702.07669, 2017. URL: <http://arxiv.org/abs/1702.07669>.
  - 19 Werner Fenchel. On conjugate convex functions. *Canad. J. Math*, 1(73-77), 1949.
  - 20 Michael L. Fredman. How good is the information theory bound in sorting? *Theor. Comput. Sci.*, 1(4):355–361, 1976. doi:10.1016/0304-3975(76)90078-5.
  - 21 Michael L. Fredman. New bounds on the complexity of the shortest path problem. *SIAM J. Comput.*, 5(1):83–89, 1976. doi:10.1137/0205006.
  - 22 Anka Gajentaan and Mark H. Overmars. On a class of  $o(n^2)$  problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995. doi:10.1016/0925-7721(95)00022-2.
  - 23 M.R. Garey and D.S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. San Francisco: Freeman, 1979.
  - 24 Isaac Goldstein, Tsvi Kopelowitz, Moshe Lewenstein, and Ely Porat. How hard is it to find (honest) witnesses? In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPICs*, pages 45:1–45:16. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPICs.ESA.2016.45.
  - 25 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.
  - 26 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.
  - 27 Klaus Jansen and Stefan E.J. Kraft. A Faster FPTAS for the Unbounded Knapsack Problem. In Zsuzsanna Lipták and William F. Smyth, editors, *Combinatorial Algorithms: 26th International Workshop, IWOCA 2015, Verona, Italy, October 5-7, 2015, Revised Selected Papers*, pages 274–286, Cham, 2016. Springer International Publishing. doi:10.1007/978-3-319-29516-9\_23.
  - 28 Hans Kellerer and Ulrich Pferschy. Improved Dynamic Programming in Connection with an FPTAS for the Knapsack Problem. *Journal of Combinatorial Optimization*, 8(1):5–11, 2004. doi:10.1023/B:JOCO.0000021934.29833.6b.
  - 29 Eduardo Sany Laber, Wilfredo Bardales Roncalla, and Ferdinando Cicalese. On lower bounds for the maximum consecutive subsums problem and the  $(\min,+)$ -convolution. In *2014 IEEE International Symposium on Information Theory, Honolulu, HI, USA, June 29 – July 4, 2014*, pages 1807–1811. IEEE, 2014. doi:10.1109/ISIT.2014.6875145.
  - 30 Yves Lucet. Faster than the Fast Legendre Transform, the Linear-time Legendre Transform. *Numerical Algorithms*, 16(2):171–185, 1997. doi:10.1023/A:1019191114493.
  - 31 Tanaeem M. Moosa and M. Sohel Rahman. Indexing permutations for binary strings. *Inf. Process. Lett.*, 110(18-19):795–798, 2010. doi:10.1016/j.ipl.2010.06.012.
  - 32 Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.
  - 33 Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 603–610. ACM, 2010. doi:10.1145/1806689.1806772.
  - 34 Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–391, June 1983. doi:10.1016/0022-0000(83)90006-5.
  - 35 Godfried Toussaint. The geometry of musical rhythm. In Jin Akiyama, Mikio Kano, and Xuehou Tan, editors, *Discrete and Computational Geometry: Japanese Conference*,

- JCDCG 2004, Tokyo, Japan, October 8-11, 2004, Revised Selected Papers*, pages 198–212, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/11589440\_20.
- 36 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.
- 37 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC'14, pages 664–673, New York, NY, USA, 2014. ACM. doi:10.1145/2591796.2591811.
- 38 Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In Thore Husfeldt and Iyad A. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, volume 43 of *LIPICs*, pages 17–29. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPICs.IPEC.2015.17.
- 39 Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM Journal on Computing*, 42(3):831–854, 2013.
- 40 Uri Zwick. All pairs shortest paths in weighted directed graphs-exact and almost exact algorithms. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 310–319. IEEE, 1998.