Tight Hardness Results for Maximum Weight Rectangles*

Arturs Backurs[†1], Nishanth Dikkala[‡2], and Christos Tzamos[§3]

1 **MIT, Cambridge, USA**
backurs@mit.edu
2 **MIT, Cambridge, USA**
ndikkala@mit.edu
3 **MIT, Cambridge, USA**
tzamos@mit.edu

Abstract

Given n weighted points (positive or negative) in d dimensions, what is the axis-aligned box which maximizes the total weight of the points it contains?

The best known algorithm for this problem is based on a reduction to a related problem, the Weighted Depth problem [Chan, FOCS, 2013], and runs in time $O(n^d)$. It was conjectured [Barbay et al., CCCG, 2013] that this runtime is tight up to subpolynomial factors. We answer this conjecture affirmatively by providing a matching conditional lower bound. We also provide conditional lower bounds for the special case when points are arranged in a grid (a well studied problem known as Maximum Subarray problem) as well as for other related problems.

All our lower bounds are based on assumptions that the best known algorithms for the All-Pairs Shortest Paths problem (APSP) and for the Max-Weight k-Clique problem in edge-weighted graphs are essentially optimal.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Maximum Rectangles, Hardness in P

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.81

1 Introduction

Consider a set of points in the plane. Each point is assigned a real weight that can be either positive or negative. The Max-Weight Rectangle problem asks to find an axis parallel rectangle that maximizes the total weight of the points it contains. This problem (and its close variants) is one of the most basic problems in computational geometry and is used as a subroutine in many applications [17, 20, 23, 7, 6]. Despite significant work over the past two decades, the best known algorithm runs in time quadratic in the number of points [16, 14, 9]. It has been conjectured that there is no strongly subquadratic time algorithm[1] for this problem [9].

An important special case of the Max-Weight Rectangle problem is when the points are arranged in a square grid. In this case the input is given as an $n \times n$ matrix filled with

* The full version of the paper is available at [8].
† A. Backurs is supported by grants from the NSF and the Simons Investigator award.
‡ N. Dikkala is supported by grants from the NSF.
§ C. Tzamos is supported by NSF Award CCF-0953960 (CAREER) and a Simons Award for Graduate Students in Theoretical Computer Science.
1 A strongly subquadratic algorithm runs in time $O(N^{2-\varepsilon})$ for constant $\varepsilon > 0$.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 81; pp. 81:1–81:13
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Table 1 Upper bounds and conditional lower bounds for the various problems studied. The bounds shown ignore subpolynomial factors.

Problem	**In 2 dimensions**	**In d dimensions**
MAX-WEIGHT RECTANGLE on N weighted points	$O(N^2)$ [9, 12] $\Omega(N^2)$ [**this work**]	$O(N^d)$ [9, 12] $\Omega(N^d)$ [**this work**]
MAXIMUM SUBARRAY on $n \times \cdots \times n$ arrays	$O(n^3)$ [32, 31] $\Omega(n^3)$ [**this work**]	$O(n^{2d-1})$ [Kadane's algorithm] $\Omega(n^{3d/2})$ [**this work**]
MAXIMUM SQUARE SUBARRAY on $n \times \cdots \times n$ arrays	$O(n^3)$ [trivial] $\Omega(n^3)$ [**this work**]	$O(n^{d+1})$ [trivial] $\Omega(n^{d+1})$ [**this work**]
WEIGHTED DEPTH on N weighted boxes	$O(N)$ [12] $\Omega(N)$ [trivial]	$O(N^{d/2})$ [12] $\Omega(N^{d/2})$ [**this work**]

real numbers and the objective is to compute a subarray that maximizes the sum of its entries [27, 31, 30, 28, 13]. This problem, known as MAXIMUM SUBARRAY problem, has applications in pattern matching [19], data mining and visualization [20] (see [31] for additional references). The particular structure of the MAXIMUM SUBARRAY problem allows for algorithms that run in $O(n^3)$, i.e. $O(N^{3/2})$ with respect to the input size $N = n^2$, as opposed to $O(N^2)$ which is the best algorithm for the more general MAX-WEIGHT RECTANGLE problem.

One interesting question is if this discrepancy between the runtimes of these two very related problems can be avoided. Is it possible to apply ideas from one to improve the runtimes of the other? Despite considerable effort there has been no significant improvement to their runtime other than by subpolynomial factors since they were originally studied.

In this work, we attempt to explain this apparent barrier for faster runtimes by giving evidence of the inherent hardness of the problems. In particular, we show that a strongly subquadratic algorithm for MAX-WEIGHT RECTANGLE would imply a breakthrough for fundamental graph problems. We show similar consequences for $O(N^{3/2-\varepsilon})$ algorithms for the MAXIMUM SUBARRAY problem. Our lower bounds are based on standard hardness assumptions for the ALL-PAIRS SHORTEST PATHS and the MAX-WEIGHT k-CLIQUE problems and generalize to the higher-dimensional versions of the problems.

1.1 Related work on the problems

In one dimension, the MAX-WEIGHT RECTANGLE problem and MAXIMUM SUBARRAY problem are identical. The 1-D problem was first posed by Ulf Grenander for pattern detection in images, and a linear time algorithm was found by Jay Kadane [10].

In two dimensions, Dobkin et al [16, 15, 24] studied the MAX-WEIGHT RECTANGLE problem in the case where weights are $+1$ or -1 for its applications to computer graphics and machine learning. They presented the first $O(N^2 \log N)$ algorithm. More recently, Cortés et al [14] studied the problem with arbitrary weights and they developed an algorithm with the same runtime applicable to many variants of the problem. An even faster algorithm was shown by Barbay et al. [9] that runs in $O(N^2)$ time.

For higher dimensions, Barbay et al [9] show a reduction to the related WEIGHTED DEPTH problem which allows them to achieve runtime $O(N^d)$. Given N axis-parallel rectangular weighted boxes, the WEIGHTED DEPTH problem asks to find a point that maximizes the total weight of all boxes that contain it. Compared to the MAX-WEIGHT RECTANGLE where we are given points and we aim to find the best box, in this problem, we are given boxes and the aim is to find the best point. The WEIGHTED DEPTH problem is also related to Klee's measure problem[2] which has a long line of research. All known algorithms for one problem can be adjusted to work for the other [12]. The Weighted Depth problem was first solved in $O(N^{d/2}\log n)$ by Overmars and Yap [26] and was improved to $O(N^{d/2})$ by Timothy M. Chan [12] who gave a surprisingly simple divide and conquer algorithm.

A different line of work, studies the Maximum Subarray problem. Kadane's algorithm for the 1-dimensional problem can be generalized in higher dimensions for d-dimensional $n \times \cdots \times n$ arrays giving $O(n^{2d-1})$ which implies an $O(n^3)$ algorithm when the array is a $n \times n$ matrix. Tamaki and Tokuyama [32] gave a reduction of the 2-dimensional version of the problem to the distance product problem implying a $O\left(\frac{n^3}{2^{\Omega(\sqrt{\log n})}}\right)$ algorithm by using the latest algorithm for distance product by Ryan Williams [34]. Tamaki and Tokuyama's reduction was further simplified by Tadao Takaoka [31] who also gave a more practical algorithm whose expected time is close to quadratic for a wide range of random data.

1.2 Our results and techniques

Despite significant work on the Max-Weight Rectangle and Maximum Subarray problems, it seems that there is a barrier in improving the best known algorithms for these problems by polynomial factors. Our results indicate that this barrier is inherent by showing connections to well-studied fundamental graph problems. In particular, our first result states that there is no strongly subquadratic algorithm for the Max-Weight Rectangle problem unless the Max-Weight k-Clique problem can be solved in $O(n^{k-\varepsilon})$ time, i.e. substantially faster than the currently best known algorithm. More precisely, we show the following:

▶ **Theorem 1.** *For any constant $\varepsilon > 0$, an $O(N^{2-\varepsilon})$ algorithm for the* Max-Weight Rectangle *problem on N weighted points in the plane implies an $O(n^{k-\varepsilon})$ algorithm for the* Max-Weight k-Clique *problem on a weighted graph with n vertices where $k = 4 \cdot \lceil \varepsilon^{-1} \rceil$.*

Our conditional lower bound generalizes to higher dimensions. Namely, we show that an $O(N^{d-\varepsilon})$ time algorithm for points in d-dimensions implies an $O(n^{k-\varepsilon})$ time algorithm for the Max-Weight k-Clique problem for $k = d^2\lceil \varepsilon^{-1} \rceil$. This matches the best known algorithm [9, 12] for any dimension up to subpolynomial factors. Therefore, because of our reduction, significant improvements in the runtime of the known upper bounds would imply a breakthrough algorithm for finding a k-clique of maximum weight in a graph.

To show this result, we embed an instance of the Max-Weight k-Clique problem to the Max-Weight Rectangle problem, by treating coordinates of the optimal rectangular box as base-n numbers where digits correspond to nodes in the maximum-weight k-clique. In the construction, we place points with appropriate weights so that the weight of any rectangular box corresponds to the weight of the clique it represents. We show that it is sufficient to use only $O(n^{\frac{k}{d}+1})$ points in d-dimensions to represent all weighted k-cliques which gives us the required bound by choosing an appropriately large k.

We also study the special case of the Max-Weight Rectangle problem in the plane where all points are arranged in a square grid, namely the Maximum Subarray problem. Our second result states that for $n \times n$ matrices, there is no strongly subcubic algorithm for the Maximum Subarray problem unless there exists a strongly subcubic algorithm for the All-Pairs Shortest Paths problem. More precisely, we show that:

[2] Klee's measure problem asks for the total volume of the union of N axis-parallel boxes in d dimensions.

▶ **Theorem 2.** *For any constant $\varepsilon > 0$, an $O(n^{3-\varepsilon})$ time algorithm for the* Maximum Subarray *problem on $n \times n$ matrices implies an $O(n^{3 - \varepsilon/10})$ time algorithm for the* All-Pairs Shortest Paths *problem.*

Combined with the fact that the Maximum Subarray problem reduces to the All-Pairs Shortest Paths problem as shown in [32, 31], our result implies that the two problems are equivalent, in the sense that any strongly subcubic algorithm for one would imply a strongly subcubic algorithm for the other.

To extend our lower bound to higher dimensions, we need to make a stronger hardness assumption based on the Max-Weight k-Clique problem. We show that an $O(n^{3d/2 - \varepsilon})$ time algorithm for the Maximum Subarray problem in d-dimensions implies an $O(n^{k-\varepsilon})$ time algorithm for the Max-Weight k-Clique problem. To prove this result, we introduce the following intermediate problem: Given a graph G find a maximum weight subgraph H that is isomorphic to a clique on $2d$ nodes without the edges of a matching (Max-Weight Clique without Matching problem). This graph H contains a large clique of size $3d/2$ as a minor and we show that this implies that no $O(n^{3d/2 - \varepsilon})$ algorithms exist for the Max-Weight Clique without Matching problem. We complete our proof by reducing the Max-Weight Clique without Matching problem to the Maximum Subarray problem in d dimensions.

We note that the best known algorithm for the Maximum Subarray problem runs in $O(n^{2d-1})$ time and is based on Kadane's algorithm for the 1-dimensional problem. It remains an interesting open question to close this gap. To improve either the lower or upper bound, it is necessary to better understand the computational complexity of the Max-Weight Clique without Matching problem.

Another related problem we consider is the Maximum Square Subarray problem: Given an $n \times n$ matrix find a maximum subarray with sides of equal length. This problem and its higher dimensional generalization can be trivially solved in $O(n^{d+1})$ runtime by enumerating over all possible combinations of the $d + 1$ parameters, i.e. the side-length and the location of the hypercube. We give a matching lower bound based on hardness of the Max-Weight k-Clique problem.

Finally, we adapt the reduction for Klee's measure problem shown by Timothy M Chan [11] to show a lower bound for the Weighted Depth problem.

Our results are summarized in Table 1, where we compare the current best upper bounds with the conditional lower bounds that we show.

The conditional hardness results presented above are for the variants of the problems where weights are arbitrary real numbers. We note that all these bounds can be adapted to work for the case where weights are either $+1$ or -1. In this case, we reduce the (unweighted) k-Clique-Detection problem[3] to each of these problems. The k-Clique-Detection problem can be solved in $O(n^{\omega\lfloor k/3\rfloor+(k \bmod 3)})$ [25] using fast matrix multiplication, where $\omega < 2.372864$ [35, 22] is the fast matrix multiplication exponent.[4] Without using fast matrix multiplication, it is not known whether a purely combinatorial algorithm exists that runs in $O(n^{k-\varepsilon})$ time for any constant $\varepsilon > 0$ and it is a longstanding graph problem. Our lower bounds can be adapted for the $+1$ / -1 versions of the problems obtaining the same runtime exponents for combinatorial algorithms as in Table 1. Achieving better exponents

[3] Given a graph on n vertices, the k-Clique-Detection problem asks whether a k-clique exists in the graph.
[4] There is a slightly faster algorithm for the case when k is not divisible by 3 [18].

for any of these problems would imply a breakthrough combinatorial algorithm for the k-Clique-Detection problem.

There is a vast collection of problems in computation geometry for which conditional lower bounds are based on the assumption of 3-SUM hardness, i.e. that the best known algorithm for the 3-SUM problem[5] can't be solved in time $O(n^{2-\varepsilon})$. This line of research was initiated by [21] (see [33] for more references). Reducing 3-SUM problem to the problems that we study seems hard if possible at all. Our work contributes to the list of interesting geometry problems for which hardness is shown from different assumptions.

1.3 Hardness assumptions

There is a long list of works showing conditional hardness for various problems based on the All-Pairs Shortest Paths problem hardness assumption [29, 36, 4, 2, 3]. Among other results, [36] showed that deciding whether a weighted graph contains a triangle of negative weight is equivalent to the All-Pairs Shortest Paths problem meaning that a strongly subcubic algorithm for the Negative Triangle problem implies a strongly subcubic algorithm for the All-Pairs Shortest Paths problem and the other way around. It is easy to show that the problem of computing the maximum weight triangle in a graph is equivalent to the Negative Triangle problem (by inverting edge-weights of the graph and doing the binary search over the weight of the max-weight triangle). Computing a max-weight triangle is a special case of the problem of computing a max-weight k-clique in a graph for a fixed integer k. This is a very well studied computational problem and despite serious efforts, the best known algorithm for this problem still runs in time $O(n^{k-o(1)})$, which matches the runtime of the trivial algorithm up to subpolynomial factors. The assumption that there is no $O(n^{k-\varepsilon})$ time algorithm for this problem, has served as a basis for showing conditional hardness results for several problems on sequences [1, 5].

2 Preliminaries

2.1 Problems studied in this work

▶ **Definition 3** (Max-Weight Rectangle problem)**.** Given N weighted points (positive or negative) in $d \geq 2$ dimensions, what is the axis-aligned box which maximizes the total weight of the points it contains?

▶ **Definition 4** (Maximum Subarray problem)**.** Given a d-dimensional array M with n^d real-valued entries, find the d-dimensional subarray of M which maximizes the sum of the elements it contains.

▶ **Definition 5** (Max-Weight Square problem)**.** Given a d-dimensional array M with n^d real-valued entries, find the d-dimensional square (hypercube) subarray of M, i.e. a rectangular box with all sides of equal length, which maximizes the sum of the elements it contains.

▶ **Definition 6** (Weighted Depth problem)**.** Given a set of N weighted axis-parallel boxes in d-dimensional space $\mathbb{R}^d$, find a point $p \in \mathbb{R}^d$ that maximizes the sum of the weights of the boxes containing p.

[5] Given a set of integers, decide if there are 3 integers that sum up to 0.

2.2 Hardness assumptions

We use the hardness assumptions of the following problems. Whenever we refer to a weighted graph, we assume that the graph is *edge-weighted* (as opposed to node-weighted).

▶ **Definition 7** (All-Pairs Shortest Paths problem). Given a weighted undirected graph $G = (V, E)$ such that $|V| = n$, find the shortest path between u and v for every $u, v \in V$.

▶ **Definition 8** (Negative Triangle problem). Given a weighted undirected graph $G = (V, E)$ such that $|V| = n$, output yes if there exists a triangle in the graph with negative total edge weight.

▶ **Definition 9** (Max-Weight k-Clique problem). Given an integer k and a weighted graph $G = (V, E)$ with n vertices, output the maximum total edge-weight of a k-clique in the graph. W.l.o.g. we assume that the graph is complete since otherwise we can set the weight of non-existent edges to be equal to a negative integer with large absolute value.

For any fixed k, the best known algorithm for the Max-Weight k-Clique problem runs in time $O(n^{k-o(1)})$.

In Sections 3 and 5, we use the following variant of the Max-Weight k-Clique problem which can be shown to be equivalent to Definition 9:

▶ **Definition 10** (Max-Weight k-Clique problem for k-partite graphs). Given an integer k and a weighted k-partite graph $G = (V_1 \cup \ldots \cup V_k,\ E)$ with kn vertices such that $|V_i| = n$ for all $i \in [k]$. Choose k vertices $v_i \in V_i$ and consider total edge-weight of the k-clique induced by these vertices. Output the maximum total-edge weight of a clique in the graph.

Notation

For any integer n, we denote the set $\{1, 2, \ldots, n\}$ by $[n]$. For a set S and an integer d, we denote the set $\{(s_1, \ldots, s_d) \mid s_i \in S\}$ by S^d.

3 Hardness of the Max-Weight Rectangle problem

The goal of this section is to show a hardness result for the Max-Weight Rectangle problem making the assumption of Max-Weight k-Clique hardness. We will show the result directly for any constant number of dimensions.

▶ **Theorem 11.** *For any constants $\varepsilon > 0$ and d, an $O(N^{d-\varepsilon})$ time algorithm for the* Max-Weight Rectangle *problem on N weighted points in d-dimensions implies an $O(n^{K-\varepsilon})$ time algorithm for the* Max-Weight K-Clique *problem on a weighted graph with n vertices for $K = d^2\lceil\varepsilon^{-1}\rceil$.*

We set $k = d \cdot \lceil\varepsilon^{-1}\rceil$. To prove the theorem, we will construct an instance of the Max-Weight Rectangle problem whose answer computes a max-weight dk-clique in a $(d \times k)$-partite weighted graph G with n nodes in each of its parts. The Max-Weight dk-Clique problem on general graphs reduces to this case since we can create $d \times k$ copies of the nodes and connect nodes among different parts with edge-weights as in the original graph.

The instance of the Max-Weight Rectangle problem will consist of $N = O(n^{k+1})$ points with integer coordinates $\{-n^k, ..., n^k\}^d$. For such an instance the required runtime for the Max-Weight Rectangle problem, from the theorem statement, would imply that the maximum weight dk-clique can be computed in

$$O\left(N^{d-\varepsilon}\right) = O\left(n^{(k+1)(d-\varepsilon)}\right) = O\left(n^{dk-\varepsilon+(d-k\varepsilon)}\right) = O\left(n^{dk-\varepsilon}\right)$$

where the last equality follows as $k \geq \frac{d}{\varepsilon}$.

To perform the reduction we introduce the following intermediate problem:

▶ **Definition 12** (Restricted Rectangle problem). Given $N = \Omega(n^k)$ weighted points in an $\{-n^k, ..., n^k\}^d$-grid, compute a rectangular box of a restricted form that maximizes the weight of its enclosed points. The rectangular box $\prod_{i=1}^{d}[-x'_i, x_i]$ must satisfy the following conditions:

1. Both $\vec{x}, \vec{x}' \in \{0, ..., n^k - 1\}^d$, and
2. Treating each coordinate x_i as a k-digit integer $(x_{i1}x_{i2}...x_{ik})_n$ in base n, i.e., $x_i = \sum_{j=1}^{k} x_{ij} n^{k-j}$ and $x_{ij} \in \{0, ..., n-1\}$, we must have $\vec{x}' = (\overline{x_d}, \overline{x_1}, \overline{x_2}, ..., \overline{x_{d-1}})$, where for an integer $z = (z_1 z_2 ... z_k)_n \in \{0, ..., n^k - 1\}$, we denote by $\overline{z} = (z_k ... z_2 z_1)_n$ the integer that has all the digits reversed.

We show that the Restricted Rectangle problem reduces to the Max-Weight Rectangle problem.

3.1 Restricted Rectangle ⇒ Max-Weight Rectangle

Consider an instance of the Restricted Rectangle problem. We can convert it to an instance of the Max-Weight Rectangle problem by introducing several additional points. Let C be a number greater than twice the sum of absolute values of all weights of the given points. We know that the solution to any rectangular box must have weight in $(-C/2, C/2)$.

The conditions of the Restricted Rectangle require that the rectangular box must contain the origin $\vec{0}$. To satisfy that we introduce a point with weight C at the origin. This forces the optimal rectangle to contain the origin since any rectangle that doesn't include this point gets weight strictly less than C.

The integrality constraint is satisfied since all points in the instance have integer coordinates so without loss of generality the optimal rectangle in the Max-Weight Rectangle problem will also have integer coordinates.

Finally, we can force $x'_2 = \overline{x_1}$, by adding for each $x_1 \in \{0, ..., n^k - 1\}$ the 4 points:

- $(x_1, -\overline{x_1}, 0, 0, .., 0)$ with weight C
- $(x_1 + 1, -\overline{x_1}, 0, 0, .., 0)$ with weight $-C$
- $(x_1, -\overline{x_1} - 1, 0, 0, .., 0)$ with weight $-C$
- $(x_1 + 1, -\overline{x_1} - 1, 0, 0, .., 0)$ with weight C

This creates $4n^k$ points and adds weight C to any rectangle with $x'_2 = \overline{x_1}$ without affecting any of the others. Working similarly for $x_2..., x_d$ we can force that the optimal solution satisfies the constraint that $\vec{x}' = (\overline{x_d}, \overline{x_1}, \overline{x_2}, ..., \overline{x_{d-1}})$.

If x and x' satisfy the conditions of the Restricted Rectangle problem, we collect weight dC for satisfying the constraints on all coordinates and C from including the point at the origin. So the total weight is at least $(d+1)C - \frac{C}{2} = (d + \frac{1}{2})C$ as every rectangle has weight at least $-C/2$ with respect to the original points. On the other hand, if at least one of the conditions is not satisfied, we receive weight strictly less than $(d + \frac{1}{2})C$. Thus, the optimal rectangular box for the Max-Weight Rectangle problem coincides with the optimal rectangular box for the Restricted Rectangle problem. The total number of points is still $O(N)$ since $N = \Omega(n^k)$ and we added $O(n^k)$ points.

3.2 Max-Weight $(d \times k)$-Partite Clique $\Rightarrow$ Restricted Rectangle

Consider a $(d \times k)$-partite weighted graph G. We label each of its parts as P_{ij} for $i \in [d]$ and $j \in [k]$. We associate each dk-clique of the graph G with a corresponding rectangular box in the RESTRICTED RECTANGLE problem. In particular, for a rectangular box defined by a point $\vec{x} \in \{0, ..., n^k - 1\}^d$, each x_{ij}, i.e. the j-th most significant digit of x_i in the base n representation, corresponds to the index of the node in part P_{ij} (0-indexed).

We now create an instance by adding points so that the total weight of every rectangular box satisfying the conditions of the RESTRICTED RECTANGLE problem is equal to the weight of its corresponding dk clique. To do that we need to take into account the weights of all the edges. We can easily take care of edges between parts $P_{11}, P_{12}, ..., P_{1k}$ of the graph by adding the following points for each $x_1 \in \{0, ..., n^k - 1\}$.

- $(x_1, 0, 0, 0, .., 0)$ with weight $W(x_1)$ equal to the weight of the k-clique $x_{11}, x_{12}, ..., x_{1k}$ in parts $P_{11}, P_{12}, ..., P_{1k}$
- $(x_1 + 1, 0, 0, 0, .., 0)$ with weight $-W(x_1)$

This creates $2n^k$ points and adds weight $W(x_1)$ to any rectangle whose first coordinate matches x_1 without affecting any of the others. We work similarly for every coordinate i from 2 through d accounting for the weight of all edges between parts P_{ia} and P_{ib} for all $i \in [d]$ and $a \neq b \in [k]$. To take into account the additional edges, we show how to add edges between parts P_{1a} and P_{2b}. For all $x_1 \in n^{k-a}\{0, ..., n^a - 1\}$ and $x_2 \in n^{k-b}\{0, ..., n^b - 1\}$ we add the points:

- $(x_1, x_2, 0, 0, .., 0)$ with weight w equal to the weight of the edge between nodes x_{1a} and x_{2b} in parts P_{1a} and P_{2b}.
- $(x_1 + n^{k-a}, x_2, 0, 0, .., 0)$ with weight $-w$
- $(x_1, x_2 + n^{k-b}, 0, 0, .., 0)$ with weight $-w$
- $(x_1 + n^{k-a}, x_2 + n^{k-b}, 0, 0, .., 0)$ with weight w

This adds weight equal to the weight of the edge between nodes x_{1a} and x_{2b} in parts P_{1a} and P_{2b} for any rectangle with corner $\vec{x}$. This creates $O(n^{a+b})$ points. This number becomes too large if $a + b > k + 1$. However, if this is the case we can instead apply the same construction in the part of the space where the numbers x_1 and x_2 appear reversed, i.e. by working with $x'_2 = \overline{x_1}$ and $x'_3 = \overline{x_2}$. For all $x'_2 \in n^{a-1}\{0, ..., n^{k+1-a} - 1\}$ and $x'_3 \in n^{b-1}\{0, ..., n^{k+1-b} - 1\}$ we add the points:

- $(0, -x'_2, -x'_3, 0, 0, .., 0)$ with weight w equal to the weight of the edge between nodes $x'_{2(k+1-a)}$ and $x'_{3(k+1-b)}$ in parts P_{1a} and P_{2b}.
- $(0, -x'_2 - n^{a-1}, -x'_3, 0, .., 0)$ with weight $-w$
- $(0, -x'_2, -x'_3 - n^{b-1}, 0, .., 0)$ with weight $-w$
- $(0, -x'_2 - n^{a-1}, -x'_3 - n^{b-1}, 0, .., 0)$ with weight w

This produces the identical effect as above creating $O(n^{2k+2-a-b})$ rectangles. If $a + b \geq k + 1$ this adds at most $O(n^{k+1})$ points as desired. We add edges between any other 2 parts $P_{i,\cdot}$ and $P_{i',\cdot}$ by performing a similar construction as above.

The overall number of points in the instance is $O(n^{k+1})$ and this completes the proof of the theorem.

4 Hardness for Maximum Subarray in 2 dimensions

In this section our goal is to show that, if we can solve the MAXIMUM SUBARRAY problem on a matrix of size $n \times n$ in time $O(n^{3-\varepsilon})$, then we can solve the NEGATIVE TRIANGLE problem in time $O(n^{3-\varepsilon})$ on n vertex graphs. It is known that a $O(n^{3-\varepsilon})$ time algorithm for the NEGATIVE TRIANGLE implies a $O(n^{3 - \varepsilon/10})$ time algorithm for the ALL-PAIRS SHORTEST

Paths problem [36]. Combining our reduction with the latter one, we obtain Theorem 2 from the introduction, which we restate here:

▶ **Theorem 2.** *For any constant $\varepsilon > 0$, an $O(n^{3-\varepsilon})$ time algorithm for the* Maximum Subarray *problem on $n \times n$ matrices implies an $O(n^{3\ -\ \varepsilon/10})$ time algorithm for the* All-Pairs Shortest Paths *problem.*

The generalization of this statement can be found in Section 5. Here we prove 2-dimensional case first because the argument is shorter.

Clearly, the Negative Triangle problem in equivalent to the Positive Triangle problem. In the remainder of this section we therefore reduce the problem of detecting whether a graph has a positive triangle to the Maximum Subarray problem.

We need the following intermediate problem:

▶ **Definition 13** (Maximum 4-Combination). Given a matrix $B \in \mathbb{R}^{m\times m}$, output

$$\max_{i,i',j,j' \in [m]\ :\ i \le i' \text{ and } j \le j'} B[i,j] + B[i',j'] - B[i,j'] - B[i',j].$$

Our reduction consists of two steps:

1. Reduce the Positive Triangle problem on n vertex graph to the Maximum 4-Combination problem on $2n \times 2n$ matrix.
2. Reduce the Maximum 4-Combination problem on $n \times n$ matrix to the Maximum Subarray matrix of size $n \times n$.

4.1 Positive Triangle ⇒ Maximum 4-Combination

Let A be the weighted adjacency matrix of size $n \times n$ of the graph and let M be the largest absolute value of an entry in A. Let $M' := 10M$ and $M'' := 100M$. We define matrix $D \in \mathbb{R}^{n\times n}$:

$$D_{i,j} = \begin{cases} M' + M'' & \text{if } i = j; \\ M'' & \text{otherwise.} \end{cases}$$

We define matrix $B \in \mathbb{R}^{2n\times 2n}$:

$$B := \begin{bmatrix} A & -A^T \\ -A^T & D \end{bmatrix}.$$

The reduction follows from the following lemma.

▶ **Lemma 14.** *Let X be the weight of the max-weight triangle in the graph corresponding to the adjacency matrix A. Let Y be the output of the* Maximum 4-Combination *algorithm when run on matrix B. The following equality holds:*

$$Y = X + M' + M''.$$

Proof. Consider integers i, j, i', j' that achieve a maximum in the Maximum 4-Combination instance as per Definition 13. Our first claim is that $i, j \le n$ and $i', j' \ge n + 1$. If this is not true, we do not collect the weight M'' and the largest output that we can get is $\le 4M' \le 9M''/10$. Note that we can easily achieve a larger output with $i = j = 1$ and $i' = j' = n + 1$.

Our second claim is that $i' = j'$. If this is not so, we do not collect the weight M' and the largest output that we can get is $M'' + 4M \leq M'' + M'/2$. Note that we can easily achieve a larger output with $i = j = 1$ and $i' = j' = n + 1$. Thus, we can denote $i' = j' = k + n$.

Now, by the construction of B, we have

$$B[i, j] + B[i', j'] - B[i, j'] - B[i', j] = A[i, j] + A[j, k] + A[k, i] + M' + M''.$$

We get the equality we need. ◀

4.2 Maximum 4-Combination ⇒ Maximum Subarray

Let $A' \in \mathbb{R}^{(n+1)\times(n+1)}$ be a matrix defined by $A'[i, j] = A[i - 1, j - 1]$ if $i, j \geq 2$ and $A'[i, j] = 0$ otherwise.

Let $C \in \mathbb{R}^{n\times n}$ be a matrix defined by $C[i, j] = A'[i, j] + A'[i + 1, j + 1] - A'[i, j + 1] - A'[i + 1, j]$.

The reduction follows from the claim that the output of the Maximum Subarray on C is equal to the output of the Maximum 4-Combination on A'. The claim follows from the following equality:

$$\sum_{i=i'}^{i''} \sum_{j=j'}^{j''} C[i, j] \;=\; A'[i'' + 1, j'' + 1] + A'[i', j'] - A'[i'' + 1, j'] - A'[i', j'' + 1].$$

The proofs of the hardness results of the next 3 sections are presented in the Appendix of this paper in the interest of space.

5 Hardness for Maximum Subarray for arbitrary number of dimensions

We state the hardness result we prove for the Maximum Subarray problem on d dimensional arrays.

▶ **Theorem 15.** *For any constant $\varepsilon > 0$, an $O\left(n^{d+\lfloor d/2\rfloor-\varepsilon}\right)$ time algorithm for the* Maximum Subarray *problem on d-dimensional array, implies an $O\left(n^{d+\lfloor d/2\rfloor-\varepsilon}\right)$ time algorithm for the* Max-Weight $(d + \lfloor d/2\rfloor)$-Clique *problem.*

The complete proof of Theorem 15 is given in the full version of the paper [8]. It generalizes the constructions used in the hardness proof of Theorem 2.

6 Hardness for Maximum Square Subarray problem

We state the hardness result we prove for the Maximum Square Subarray problem on d dimensional arrays.

▶ **Theorem 16.** *For any constant $\varepsilon > 0$, an $O\left(n^{d+1-\varepsilon}\right)$ time algorithm for the* Maximum Square Subarray *problem on a d-dimensional array implies an $O\left(n^{d+1-\varepsilon}\right)$ time algorithm for the* Max-Weight $(d + 1)$-Clique *problem.*

The proof of Theorem 16 is given in the full version of the paper [8].

7 Hardness for Weighted Depth problem

We state the hardness result we prove for the WEIGHTED DEPTH problem in d dimensional space.

▶ **Theorem 17.** *For any constant $\varepsilon > 0$, an $O\left(n^{\lfloor d/2 \rfloor - \varepsilon}\right)$ time algorithm for the* WEIGHTED DEPTH *problem in d dimensional space implies an $O\left(n^{d-2\varepsilon}\right)$ time algorithm for the* MAX-WEIGHT (d)-CLIQUE *problem.*

The proof of the above theorem is given in the full version of the paper [8].

Acknowledgements. We thank Piotr Indyk for providing helpful discussions and comments on an earlier version of the paper. We thank Linda Lynch for copy-editing parts of the paper. One of the authors, Nishanth Dikkala, would like to thank Namratha Dikkala for a discussion about efficient algorithms for geometric problems which led to some of questions pursued in this paper.

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is valiant's parser. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 98–117. IEEE, 2015.

2 Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, apsp and diameter. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1681–1697. SIAM, 2015.

3 Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 41–50. ACM, 2015.

4 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 434–443. IEEE, 2014.

5 Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Automata, Languages, and Programming*, pages 39–51. Springer, 2014.

6 Deepak Agarwal, Jeff M Phillips, and Suresh Venkatasubramanian. The hunting of the bump: on maximizing statistical discrepancy. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1137–1146. Society for Industrial and Applied Mathematics, 2006.

7 Jonathan Backer and J Mark Keil. The mono-and bichromatic empty rectangle and square problems in all dimensions. In *LATIN 2010: Theoretical Informatics*, pages 14–25. Springer, 2010.

8 Arturs Backurs, Nishanth Dikkala, and Christos Tzamos. Tight hardness results for maximum weight rectangles. *arXiv preprint arXiv:1602.05837*, 2016. URL: `http://arxiv.org/abs/1602.05837`.

9 Jérémy Barbay, Timothy M Chan, Gonzalo Navarro, and Pablo Pérez-Lantero. Maximum-weight planar boxes in o $\left(n^2\right)$ time (and better). *Information Processing Letters*, 114(8):437–445, 2014.

10 Jon Bentley. Programming pearls: algorithm design techniques. *Communications of the ACM*, 27(9):865–873, 1984.

11 Timothy M Chan. A (slightly) faster algorithm for klee's measure problem. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 94–100. ACM, 2008.

12 Timothy M Chan. Klee's measure problem made easy. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 410–419. IEEE, 2013.

13 Chih-Huai Cheng, Kuan-Yu Chen, Wen-Chin Tien, and Kun-Mao Chao. Improved algorithms for the k maximum-sums problems. In *Algorithms and Computation*, pages 799–808. Springer, 2005.

14 C Cortés, José Miguel Díaz-Báñez, Pablo Pérez-Lantero, Carlos Seara, Jorge Urrutia, and Inmaculada Ventura. Bichromatic separability with two boxes: a general approach. *Journal of Algorithms*, 64(2):79–88, 2009.

15 David P Dobkin and Dimitrios Gunopulos. Computing the rectangle discrepancy. In *Proceedings of the tenth annual symposium on Computational geometry*, pages 385–386. ACM, 1994.

16 David P Dobkin, Dimitrios Gunopulos, and Wolfgang Maass. Computing the maximum bichromatic discrepancy, with applications to computer graphics and machine learning. *journal of computer and system sciences*, 52(3):453–470, 1996.

17 Jonathan Eckstein, Peter L Hammer, Ying Liu, Mikhail Nediak, and Bruno Simeone. The maximum box problem and its application to data analysis. *Computational Optimization and Applications*, 23(3):285–298, 2002.

18 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1):57–67, 2004.

19 Paul Fischer, Klaus-U Höffgen, Hanno Lefmann, and Tomasz Luczak. Approximations with axis-aligned rectangles. In *Fundamentals of Computation Theory*, pages 244–255. Springer, 1993.

20 Takeshi Fukuda, Yasukiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. *ACM SIGMOD Record*, 25(2):13–23, 1996.

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

22 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation*, pages 296–303. ACM, 2014.

23 Ying Liu and Mikhail Nediak. Planar case of the maximum box and related problems. In *CCCG*, pages 14–18, 2003.

24 Wolfgang Maass. Efficient agnostic pac-learning with simple hypothesis. In *Proceedings of the seventh annual conference on Computational learning theory*, pages 67–75. ACM, 1994.

25 Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

26 Mark H Overmars and Chee-Keng Yap. New upper bounds in klee's measure problem. *SIAM Journal on Computing*, 20(6):1034–1045, 1991.

27 Kalyan Perumalla and Narsingh Deo. Parallel algorithms for maximum subsequence and maximum subarray. *Parallel Processing Letters*, 5(03):367–373, 1995.

28 Ke Qiu and Selim G Akl. Parallel maximum sum algorithms on interconnection networks. In *Proceedings of the Eleventh IAESTED Conference on Parallel and Distributed Computing and Systems*, pages 31–38. Citeseer, 1999.

29 Liam Roditty and Uri Zwick. On dynamic shortest paths problems. In *Algorithms–ESA 2004*, pages 580–591. Springer, 2004.

30 Douglas R Smith. Applications of a strategy for designing divide-and-conquer algorithms. *Science of Computer Programming*, 8(3):213–229, 1987.

31 Tadao Takaoka. Efficient algorithms for the maximum subarray problem by distance matrix multiplication. *Electronic Notes in Theoretical Computer Science*, 61:191–200, 2002.

32 Hisao Tamaki and Takeshi Tokuyama. Algorithms for the maxium subarray problem based on matrix multiplication. In *SODA*, volume 1998, pages 446–452, 1998.

33 Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 43. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

34 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 664–673. ACM, 2014.

35 Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 887–898. ACM, 2012.

36 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 645–654. IEEE, 2010.