Hardness for Maximum Weight Rectangles

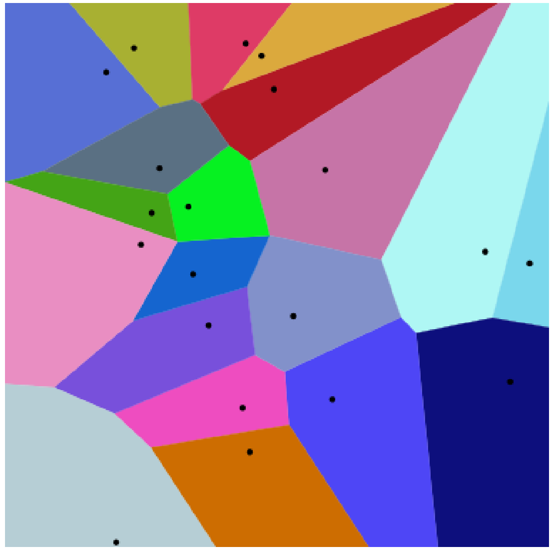
ARTURS BACKURS, MIT

NISHANTH DIKKALA, MIT

CHRISTOS TZAMOS, MIT

Motivation

- Problems in P: Common in practice.
  - String matching problems, Computational geometry etc.
- In practice an $O(n^3)$ vs $O(n^2)$ has significant difference. And $O(n^{100})$ perhaps impractical.
- If no improvements achievable, can we show lower bounds?
- Unconditional lower bounds seem out of reach. Can we give evidence that certain problems are hard?



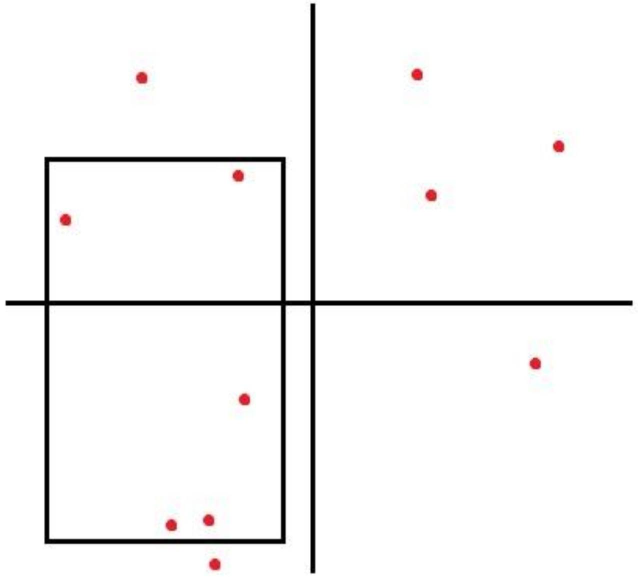
Max Weight Rectangle

INPUT N weighted points in the plane (positive or negative weights)

OUTPUT The axis-aligned rectangle which maximizes sum of weights of points enclosed.

APPLICATIONS [EHL+02, FMMT96, LN03, BK10,APV06]

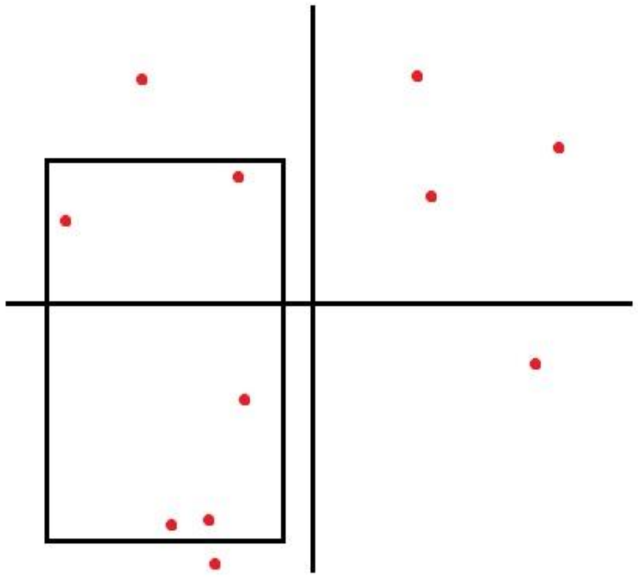
- Bichromatic Discrepancy – PAC Learning
- Data mining
- Graphics



Max Weight Rectangle

ALGORITHMS:

- **$O(N^5)$** – Naïve algorithm. Choose 4 points lying on the edges $O(N^4)$. $O(N)$ time to count the total weight inside.
- **$O(N^3)$** – Dynamic Programming
- **$O(N^2 \log N)$** – [CDBPL+09]
- **$O(N^2)$** – [BCNP14]
- **Can we get better than $O(N^2)$?**



Max Subarray – A related problem

INPUT n × n matrix with real entries,

OUTPUT The subarray of the matrix which maximizes the sum of its elements.

Applications in pattern matching, data mining [FHLL93], [FMMT96].

Special case of Max Weight Rectangle when all points lie on a grid.

Input size $N = n^2$. DP approach gives $O(N^{3/2})$.

No polynomial improvement since.

-3	5	-13	-6
40	-23	-5	-12
3	24	7	5
-42	3	4	-23

Comparison of Max Weight Rectangle and Max Subarray

MAXIMUM WEIGHT RECTANGLE

Best runtime – **$O(N^2)$**

No polynomial improvement since [BarbayChanNavarroPerez-Lantero14]

MAXIMUM SUBARRAY

Best runtime – **$O(N^{3/2})$**

Runtime obtained via a simple DP algorithm [Kadane84]. No polynomial improvement since.

Can discrepancy in runtimes of the two problems be avoided?

Questions

Can we do better than N^2 for Max Weight Rectangle?

Can we do better than $N^{3/2}$ for Max Subarray?

No! – unless breakthrough in graph theory

Hardness in P

Goal: Show hardness for problems in P analogous to NP-hardness.

Hardness Conjecture

All Pairs Shortest Paths (APSP): Given a weighted graph G, find distance between every pair of vertices.

Despite long line of research no $O(n^{3-\varepsilon})$ algorithm known for APSP.

APSP Conjecture: For any $\varepsilon>0$, there is no $O(n^{3-\varepsilon})$ time algorithm for APSP.

Author	Runtime for APSP	Year
Fredman	$n^3 \log\log^{1/3} n / \log^{1/3} n$	1976
Takaoka	$n^3 \log\log^{1/2} n / \log^{1/2} n$	1992
Dobosiewicz	$n^3/\sqrt{\log n}$	1992
Zwick	$n^3 \log\log^{1/2} n / \log n$	2004
Chan	$n^3/\log n$	2005
Han, Takaoka	$n^3 \log\log n / \log^2 n$	2012
Williams	$n^3/\exp(\sqrt{\log n})$	2014

Hardness Conjecture

APSP Conjecture: For any $\varepsilon>0$, there is no $O(n^{3-\varepsilon})$ time algorithm for APSP.

Standard conjecture used in long list of works [RoddityZwick04, WilliamsWilliams10, Abboud Williams14, AbboudGrandoniWilliams15, AbboudWilliamsYu15].

[WilliamsWilliams10] **APSP ≡ Max Weight Triangle.**
Max Weight Triangle: Given a weighted graph G, find a triangle of maximum weight.

Generalization of Max Weight Triangle is Max Weight K-clique for constant K.

Max Weight K-Clique: Given an edge weighted graph G, find the max weight clique of size K. (K is a constant)

No algorithm polynomially faster than naïve one of $O(n^K)$.

Hardness Conjecture

K-Clique Conjecture: For any ε>0 and any constant K, there is no $O(n^{K-\varepsilon})$ time algorithm for Max Weight K-clique.

Conjecture used in [AbboudBackursWilliams15], [AbboudWeimannWilliams14], [BackursTzamos16].

We will show hardness based on APSP and K-Clique conjectures.

Main Results

Theorem 1: No $O(N^{2-\varepsilon})$ algorithm exists for Max Weight Rectangle assuming K-Clique conjecture for $K=4/\varepsilon$.

Theorem 2: No $O(n^{3-\varepsilon})$ algorithm exists for Max Subarray on an $n \times n$ matrix assuming APSP conjecture.

Main Results

<u>Theorem 1:</u> No $O(N^{2-\varepsilon})$ algorithm exists for Max Weight Rectangle assuming K-Clique conjecture for $K=4/\varepsilon$.

<u>Theorem</u> 2: No $O(n^{3-\varepsilon})$ algorithm exists for Max Subarray on an $n \times n$ matrix assuming APSP conjecture.

Proof of Theorem 1

Theorem 1: Assuming Max Weight K-Clique hardness for $K=4/\varepsilon$, no $O(N^{2-\varepsilon})$ algorithm exists for Max Weight Rectangle.

Let $k = K/2=2/\varepsilon$

Graph with n vertices

Max weight K-clique

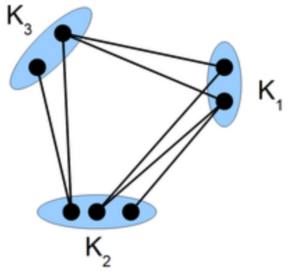
On $O(n^{k+1})$ points

Maximum Weight Rectangle

$$O(N^{2-\varepsilon}) = O(n^{(k+1)(2-\varepsilon)}) = O(n^{2k-\varepsilon})$$

Proof Sketch

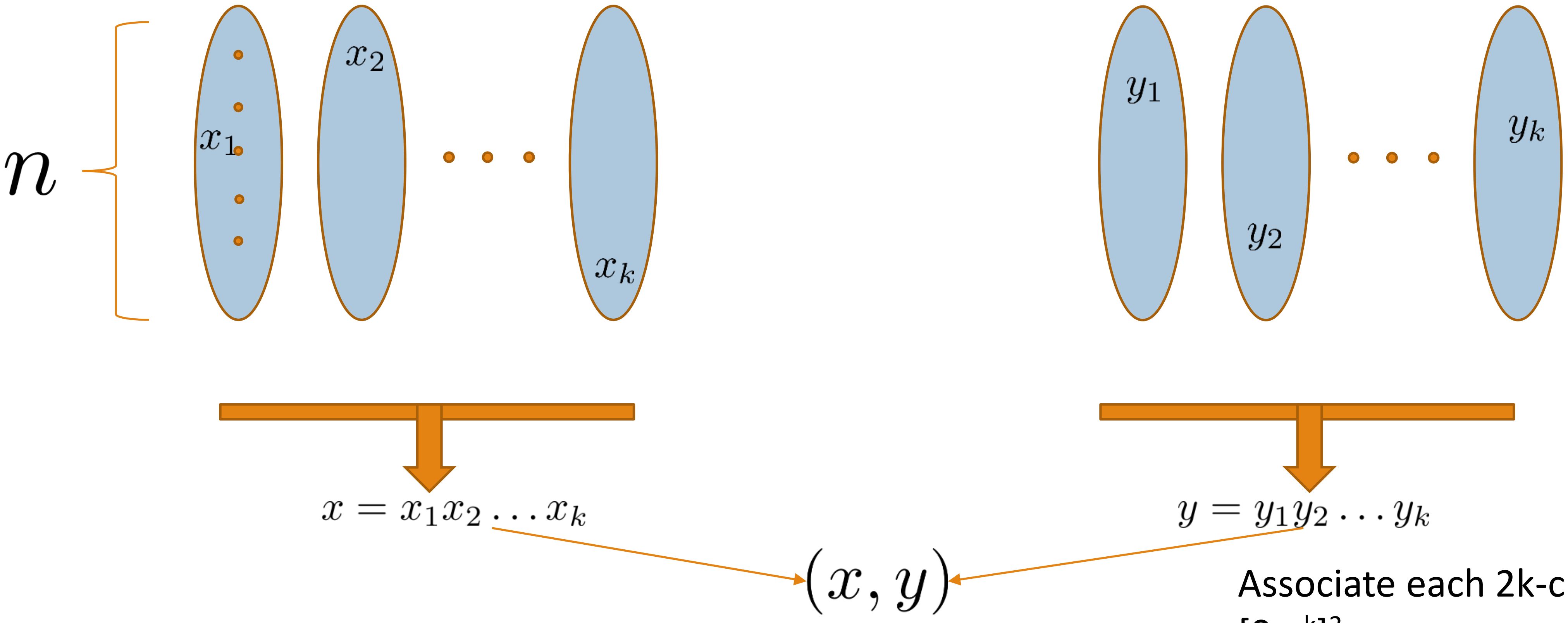
Map cliques to points on plane → Clique weights to Rectangle weights using Gadgets → Efficient Compression



Reduction from a K-partite graph.
K-clique on general graphs equivalent to K-clique on K-partite graphs.

Proof Idea 1 – Mapping cliques to points

2k = K

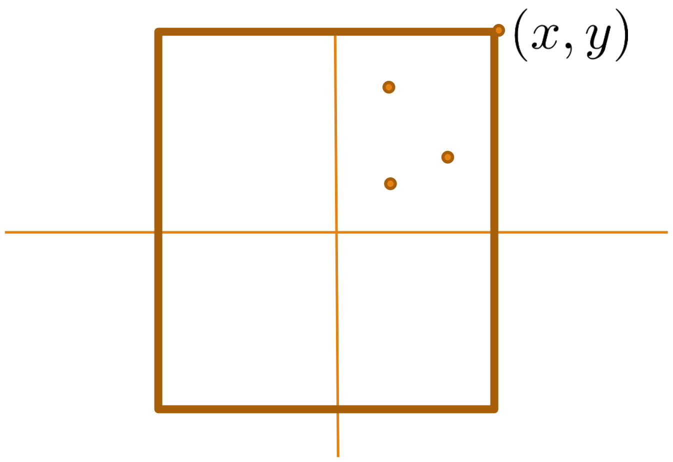


Associate each 2k-clique to a point in $[0,n^k]^2$

Proof Idea 2 – Clique Weights to Rectangle Weights

Add large weight at origin OPT includes origin always. Will have one corner in each quadrant.

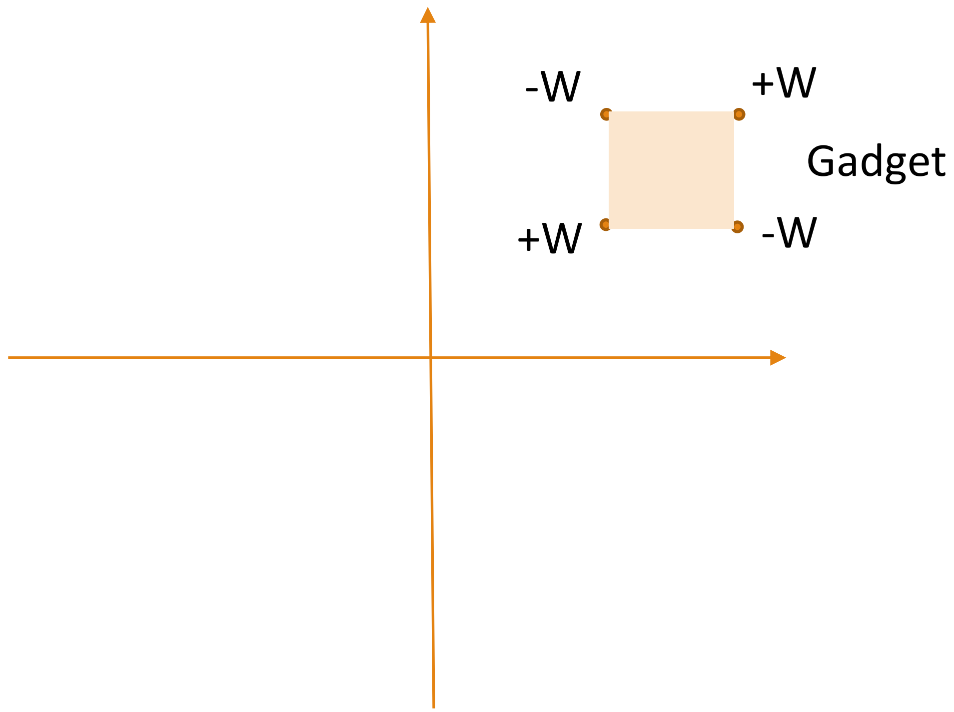
Idea: At each point (x,y) in 1st quadrant, associate weight of clique it represents.
That is, rectangle with (x,y) as top right corner gets total weight = weight of clique (x,y).



Gadget for localizing weight contribution

Gadget for weight W: Add 4 points as shown.

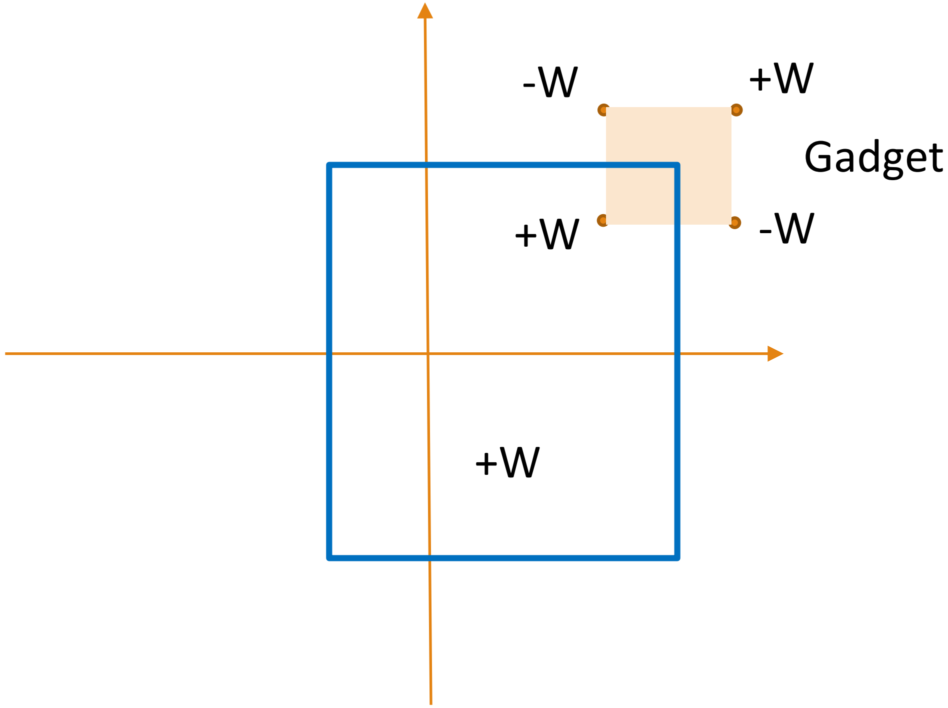
Ensures any rect. with top right corner in shaded region gets +W, else gets a total weight of 0 from this gadget.



Gadget for localizing weight contribution

Gadget for weight W: Add 4 points as shown.

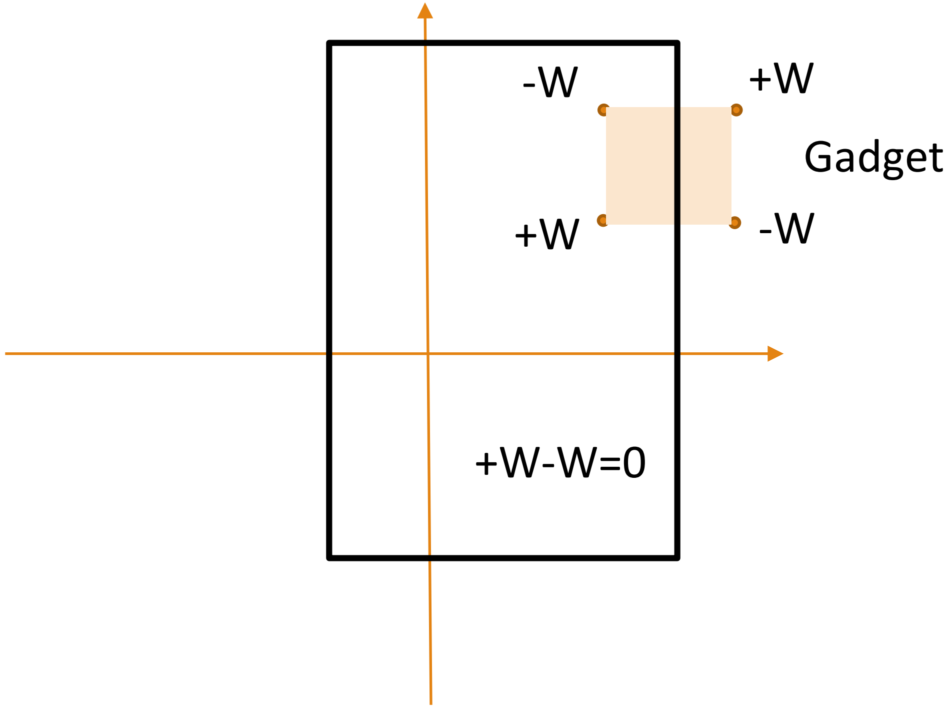
Ensures any rect. with top right corner in shaded region gets +W, else gets a total weight of 0 from this gadget.



Gadget for localizing weight contribution

Gadget for weight W: Add 4 points as shown.

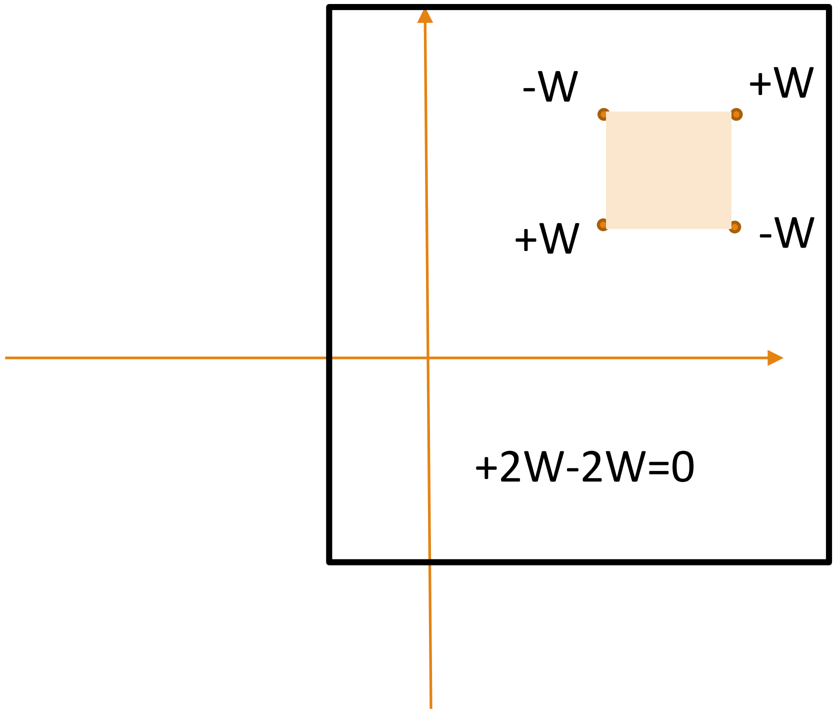
Ensures any rect. with top right corner in shaded region gets +W, else gets a total weight of 0 from this gadget.



Gadget for localizing weight contribution

Gadget for weight W: Add 4 points as shown.

Ensures any rect. with top right corner in shaded region gets +W, else gets a total weight of 0 from this gadget.



Proof idea 2 – Clique Weights to Rectangle Weights

Idea: Construct gadget at each point with the weight of the clique associated with the point.

Issue: Too many points!! $O(n^{2k})$

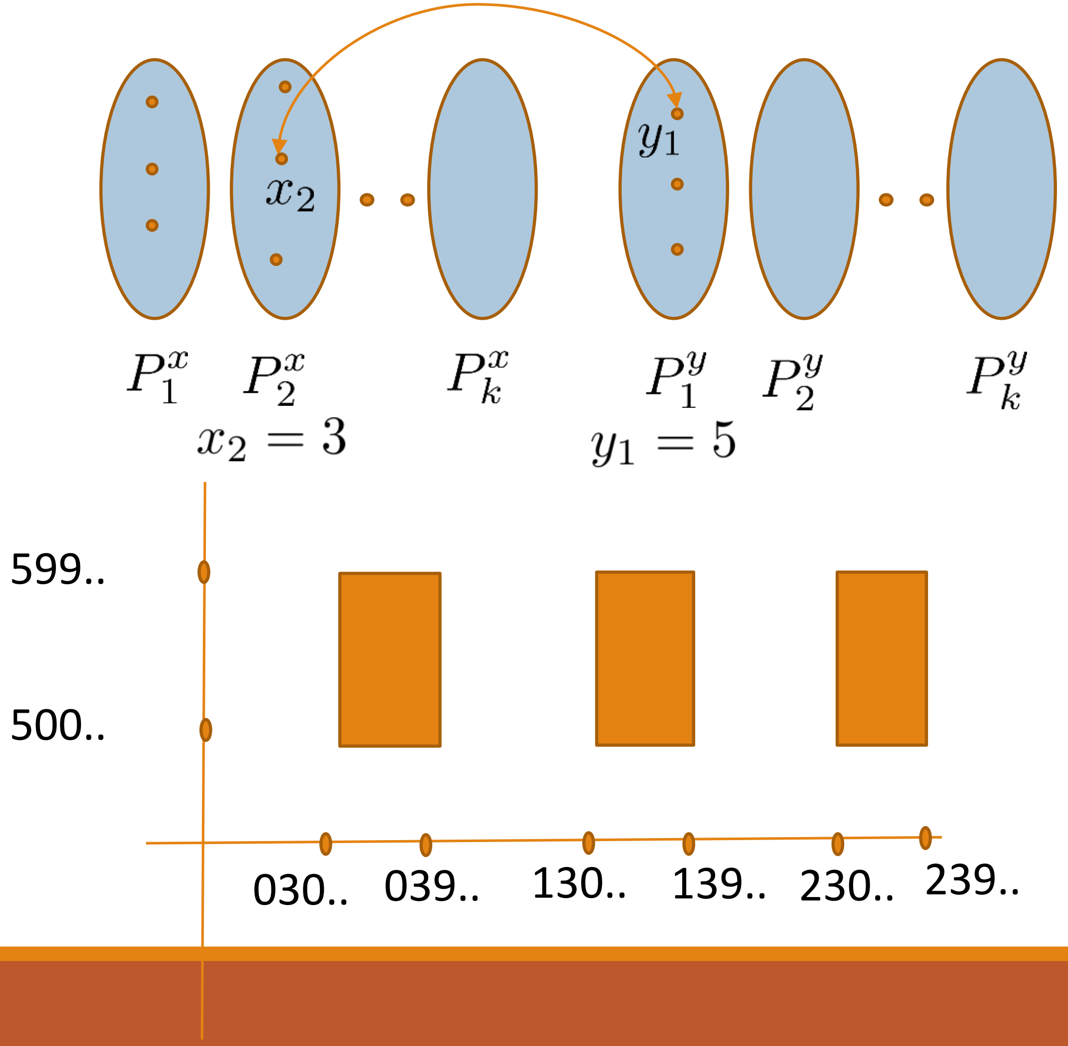
Recall: Have a budget of $O(n^{k+1})$ points.

Idea: View clique weight as sum of edge weights. Handle contribution from each edge using $O(n^{k-1})$ points.

Q. What points in plane should get contribution of a given edge?

A. Those points which represent cliques which include given edge.

Proof Idea – Handling Edge contribution



For eg., take edge between vertex 3 in P_2^x and vertex 5 in P_1^y.

All points in shaded regions represent cliques which include this edge.

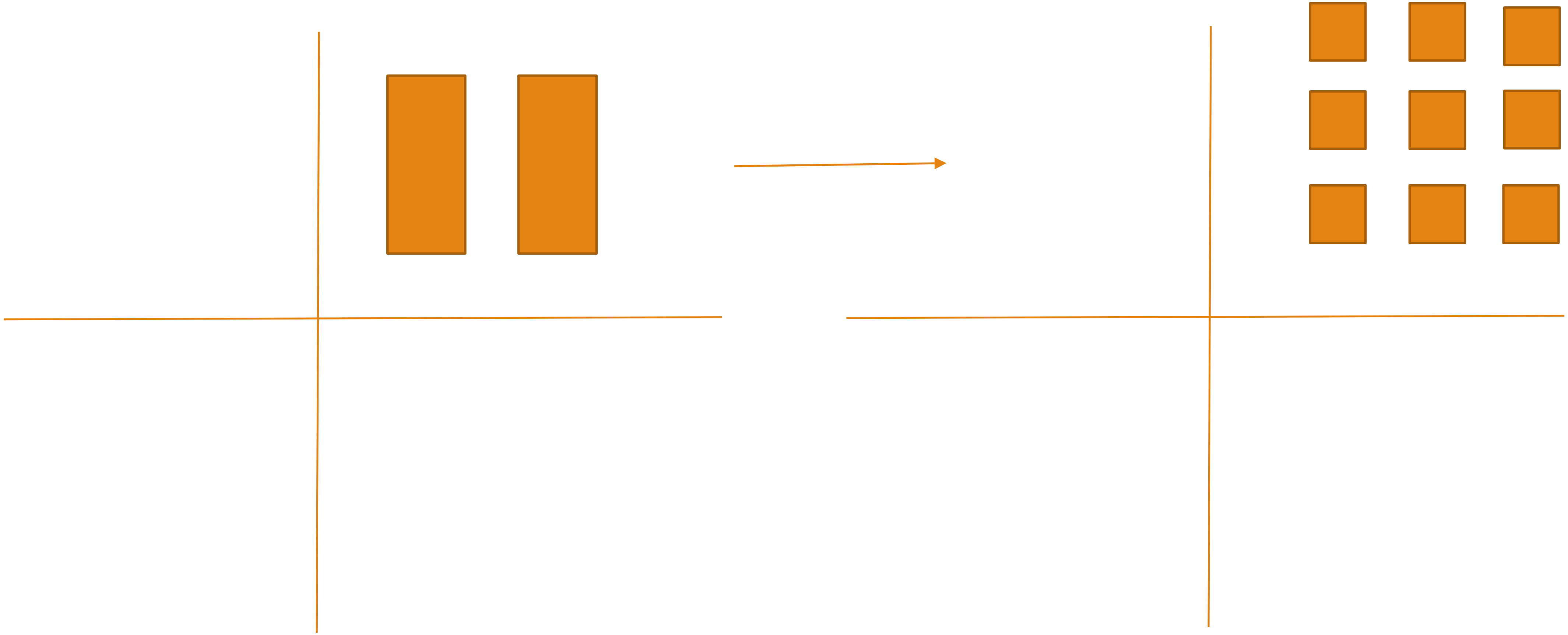
One gadget per shaded region. O(n) points for this edge. For all edges between P_2^x and P_1^y : use $O(n^3)$ points.

Edges between partitions P_i^x and P_j^y using $O(n^{i+j})$ points.

Too expensive when i+j > k+1 !

Budget of $O(n^{k+1})$

Proof Idea – Handling Edge Contribution

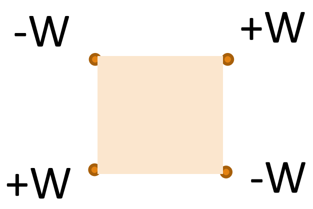


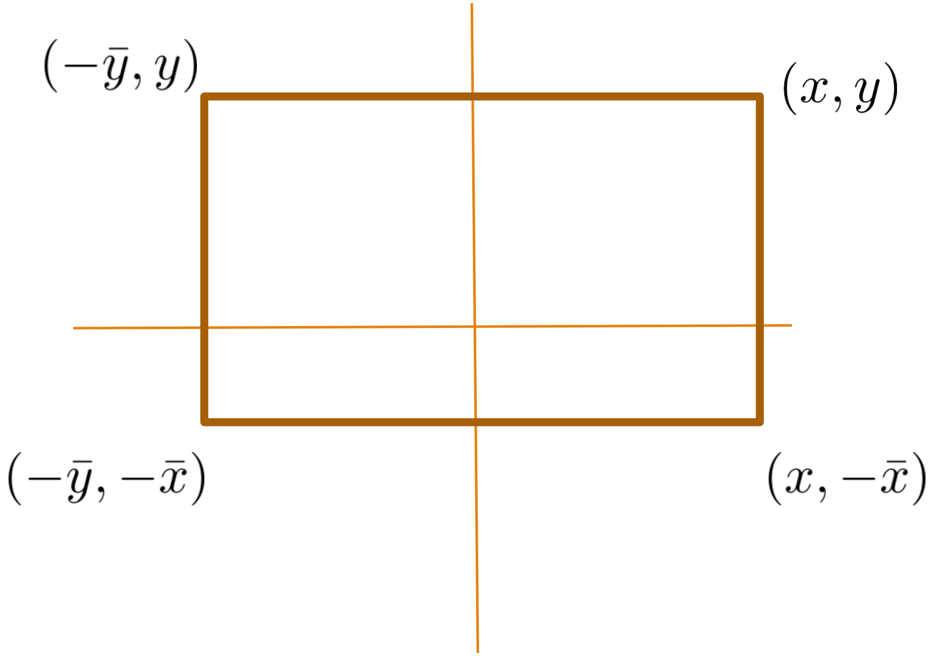
Proof Idea 3 – Exploiting the 3rd quadrant

So far used only top right corner. For edges whose representative regions are too many in 1st quadrant, use bottom left corner.

Define $\bar{x}$ to be the k digit number which has the digits of x in **reverse**.

For eg., if $x = 123$, then $\bar{x} = 321$





Using ***gadget*** described, can ensure in OPT
Top right is (x, y) $\iff$ Bottom left is $(-\bar{y}, -\bar{x})$

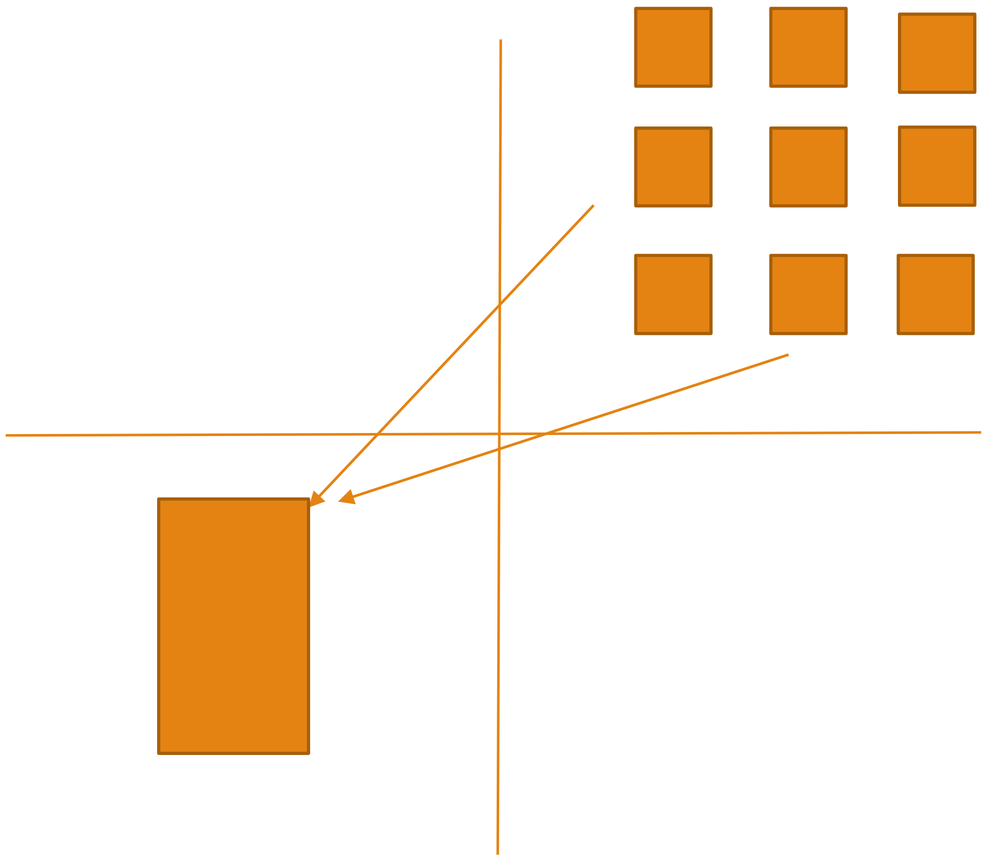
Proof Idea 3 – Exploiting the 3rd quadrant

Now points in Q3 also associated to cliques.

Do construction in Q3 for edges which have too many associated regions in Q1

Requires only $O(n^{k+1})$ points. □

Algorithms for Max Weight Rectangle and Max Subarray are probably optimal.



Summary - Table of Results

Bounds shown ignore subpolynomial factors.

Problem	In 2-dimensions	In d-dimensions
Maximum Weight Rectangle On N weighted points	$O(N^2)$ - [BCNP 14, Chan13] $\Omega(N^2)$	$O(N^d)$ - [BCNP 14, Chan13] $\Omega(N^d)$
Maximum Subarray On n × … × n arrays	$O(n^3)$ - [TT98,Tak02] $\Omega(n^3)$	$O(n^{2d-1})$ - [Kadane84] $\Omega(n^{3d/2})$
Maximum Square Subarray On n × … × n arrays	$O(n^3)$ - Trivial $\Omega(n^3)$	$O(n^{d+1})$ - Trivial $\Omega(n^{d+1})$

THANK YOU Questions?