Exact adaptive parallel algorithms for data depth problems

Vera Rosta
Department of Mathematics and Statistics
McGill University, Montreal

joint work with

Komei Fukuda
School of Computer Science
McGill University, Montreal

Data Depth is an Interdisciplinary Problem

Equivalent formulations, partial solutions, exploiting results in different fields are necessary to solve it. The fields I will refer to as necessary are:

Statistics
Non-parametric Statistics

Convex Geometry
Computational Geometry
Combinatorial Abstraction of Geometry

Complexity Theory

Combinatorial Optimization
Mathematical Programming

High-dimensional Geometric Computing
Parallel Computing
Implementation, Software

Subject of Statistics: **measures of central location, ranks**

Non-parametric Statistics, data analysis without assumptions about the underlying probability distribution.

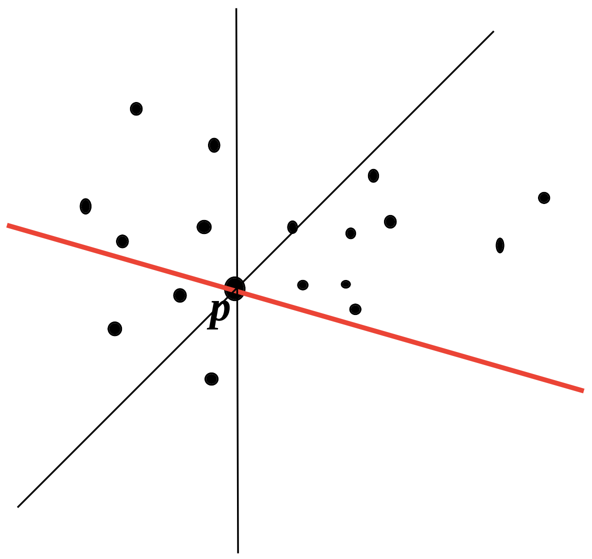
Different notions of data depths (location) by
Tukey (1975), Oja (1983), Liu (1990), Donoho and Gasko (1992), Singh (1992), Rousseeuw and Hubert (1999)

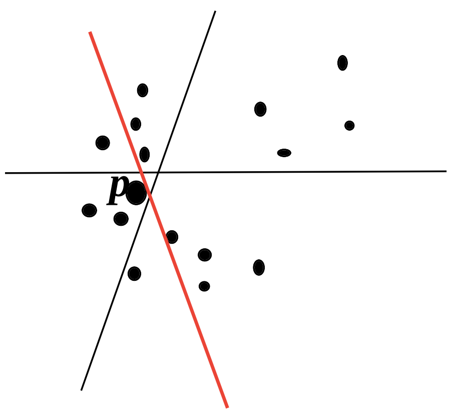
Since non-parametric methods are distribution free, instead of probability measures, these new location measures are often **geometrical**.

For multidimensional data this becomes a non-trivial **Geometric Computation problem**.

Not enough to have theoretical algorithms, **Implementations are necessary**, to be able to compute **confidence intervals, hypothesis tests** etc, based on these new rank and location measures.

Given a set $S \subset \mathbb{R}^d$ of n datapoints, the **location depth** (Tukey 1975) of a point p relative to S: **min number of data points in any halfspace containing** p.





Tukey median: a point with **max possible depth**.

Efficient algorithms exist if $d = 2$, for computing the set of points with location depth at least k, the k depth regions, and the Tukey median.
(Matousek, Shamir, Steiger, Langerman, Millner, Streinu, Rousseeuw, Struyfs, Naor).
Some of these results are extended to $d = 3$.

There are no implemented deterministic algorithms to compute location depths related measures in higher than 3 dimensions.

Complexity:

Johnson and Preparata (1978): **Computation of the location depth of a given point is NP-complete**.
(There is no hope for polynomial-time algorithm)

Amaldi-Kahn (1995): It is NP-hard to approximate the location depth of a point. (This problem does not admit a polynomial-time approximation scheme, unless P=NP)

Hardness results are related to the worst case, (in actual problems it might be rare).

The complexity of adaptive algorithms depend on the problem at hand, more relevant for statistical computations.

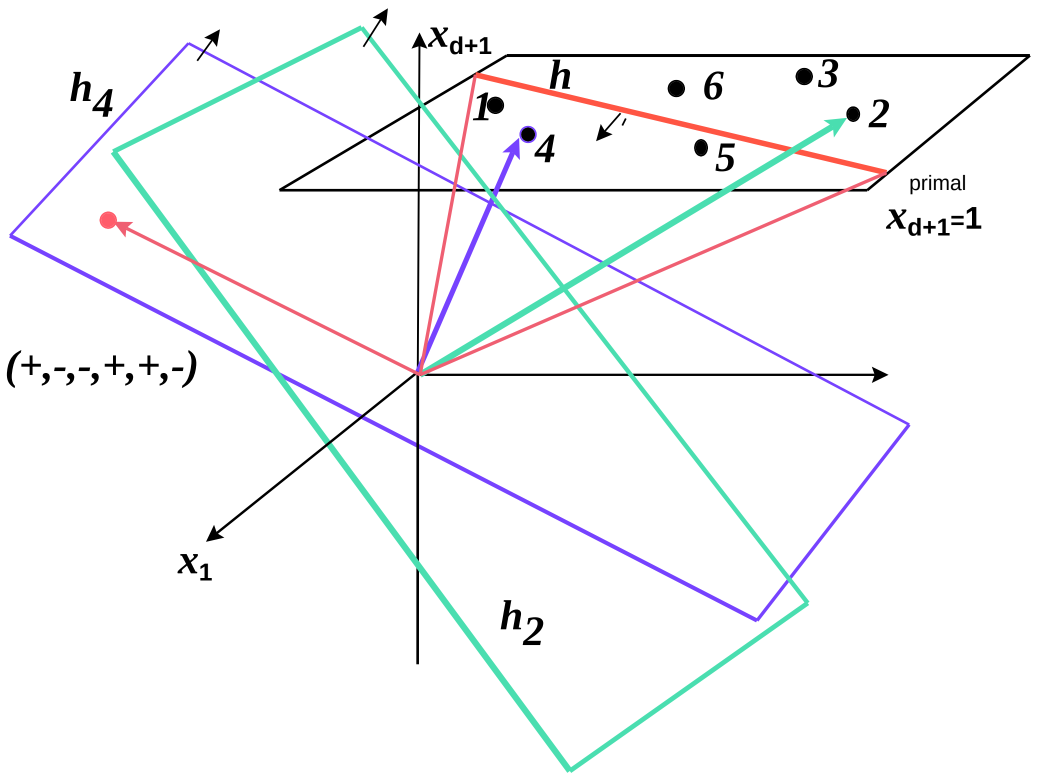
We present the **first adaptive, deterministic, highly parallelizable algorithms in any dimension** to

a) **compute the location depth of a point** in $\mathbb{R}^d$,

b) **construction of the location depths regions boundary,**

using a new **memory efficient, output sensitive, highly paralellizable enumeration algorithm for hyperplane arrangements**, which is being **modified to be adaptive**, to reduce complexity depending on the problem at hand.

Data $S = \{p^1, p^2, \ldots, p^n\}$ in $\mathbb{R}^d$. Embedd in $\mathbb{R}^{d+1}$.
Primal: $\hat{p}^i = (p^i, 1) \in \mathbb{R}^{d+1}, \quad i = 1, 2, \ldots, n$
Dual arrangement A_S**:** $< \hat{p}^i, \hat{x} >= 0, \quad i = 1, 2, \ldots, n$.



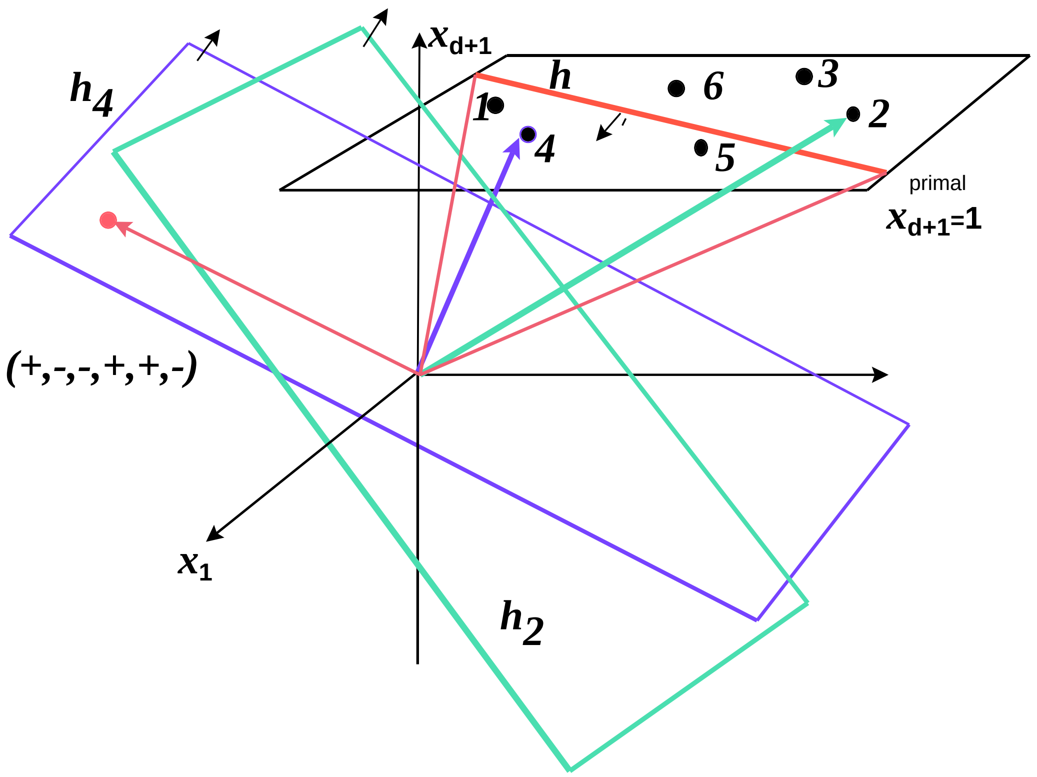
For any $\hat{x} \in \mathbb{R}^{d+1}$, define the *sign vector* $\sigma(\hat{x}) \in \{+, 0, -\}^n$ as the vector whose ith component is the sign of $< \hat{p}^i, \hat{x} >$.

$\mathcal{F}_S = \{\sigma(\hat{x}) : \hat{x} \in \mathbb{R}^{d+1}\}$, (faces of the arrangement A_S).

$\mathcal{C}_S$: set of *cells*, (full dimensional faces, sign vectors with no zero components).

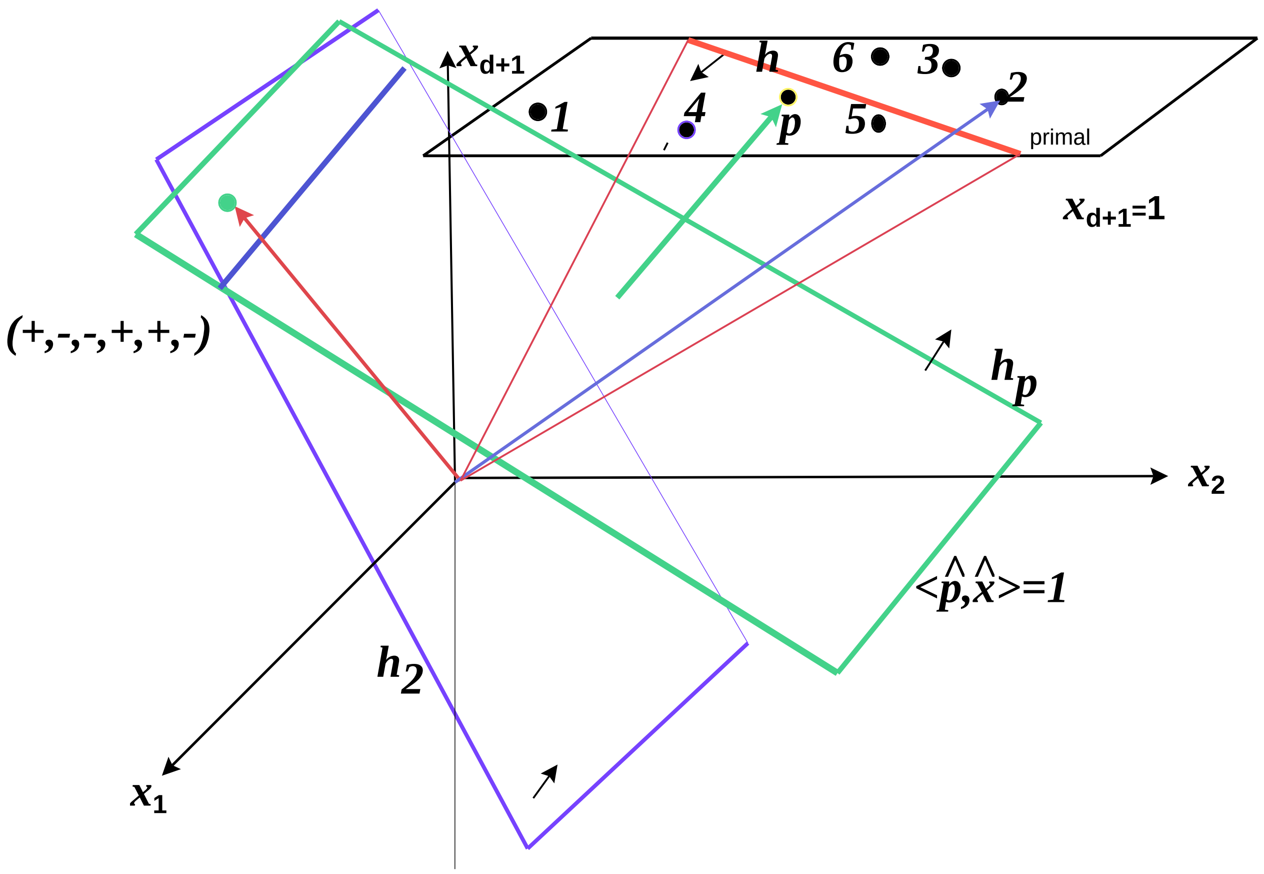
The following is the basic duality.

Proposition 0.1 *For each $X \in \{+, 0, -\}^n$, $X \in \mathcal{F}_S$ if and only if there is an oriented hyperplane h in $\mathbb{R}^d$ such that $X^+ \subseteq h^+$, $X^0 \subseteq h$ and $X^- \subseteq h^-$.*



For any sign vector X,
X^+ the *positive support* $\{i : X_i = +\}$, *zero support* X^0, *negative support* X^- and *support* $\underline{X} := X^+ \cup X^-$.
Remark: $X \in \mathcal{F}_S$ iff $-X \in \mathcal{F}_S$, where
$-X^+ = X^-, -X^- = X^+, -X^0 = X^0$.

$\mathcal{F}_S^p = \{\sigma(\hat{x}) : \hat{x} \in \mathbb{R}^{d+1}$ and $<\hat{p}, \hat{x}>= 1\}$,
$A_S^p = A_S \cap <\hat{p}, \hat{x}>= 1$, $\mathcal{C}_S^p$ the set of cells of A_S^p.



Computing the location depth of a point is equivalent to finding a cell sign vector in the hyperplane arrangement A_S^p with smallest number of positive signs.

(For each $X \in \{+, 0, -\}^n$, $X \in \mathcal{F}_S^p$ if and only if there is an oriented hyperplane h in $\mathbb{R}^d$ such that
$X^+ \cup \{p\} \subseteq h^+$, $X^0 \subseteq h$ and $X^- \subseteq h^-$.)

Theorem: *There is a search algorithm of time complexity* $O(n\ \ LP(n,d)\ |\mathcal{C}^p_S|)$ *and space complexity* $O(n\ d)$ *that computes* $\mathcal{C}^p_S$ *for any given* S *and* p*.*

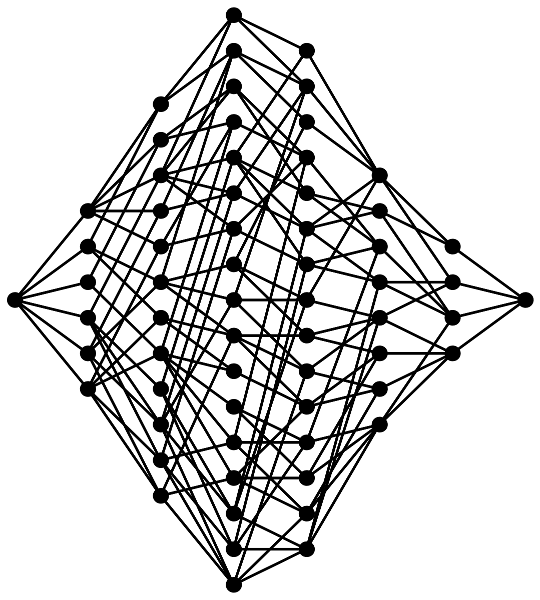
This is a modified version of a reverse search algorithm of Avis-Fukuda with an improved complexity. Here, $LP(n,d)$ denotes the time to solve an LP with n inequalities in d variables.

Hyperplane arrangement algorithms have been used in the past for data depth problems: 1) Johnson and Preparata worked with hyperplane arrangement.

2) In computational geometry there is a so-called standard hyperplane arrangement enumeration algorithm by Edelsbrunner et al., that has been used to design algorithm for computing regression depth (Mitchell-Sharir).

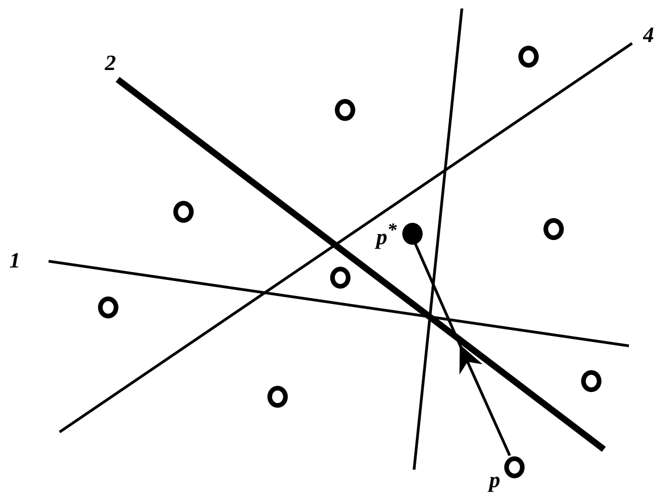
These have not been implemented, probably because the memory requirement is $O(n^d)$, Let C **be any cell known at the beginning of computation**. (Can take a random point in $\mathbb{R}^d$ and check its signature, or first using some heuristic to get a locally best initial cell.)

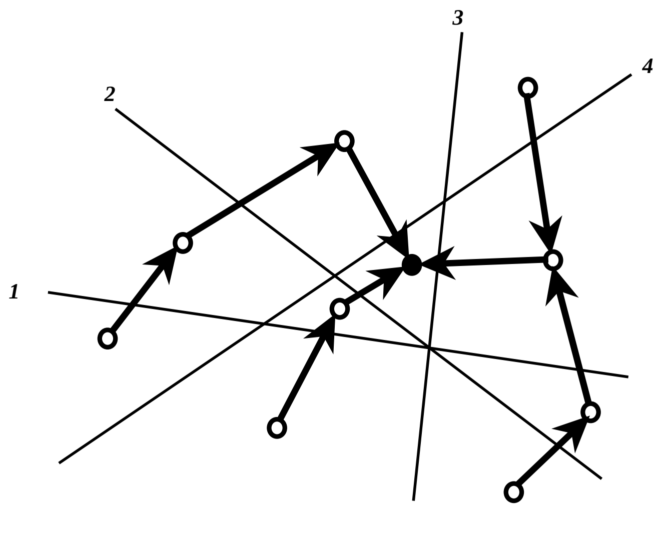
The **algorithm** is defined by two functions:
1) *adjacency oracle function* Adj: **determines the neighbor cells of any given cell** X.
Cell Graph (3 dimension, 7 hyperplanes, 64 cells)



The *adjacency oracle* $Adj(X, j)$, index $j \in \{1, \ldots, n\}$, returns the new cell if j is flippable, and NULL otherwise, using the stored LP of size $n \times d$.
(j is *flippable* in X if $X_j \neq 0$ and the vector obtained from X by reversing the jth sign is again a cell.)

2) *Local Search Function* f, that **maps any cell** X **in** $\mathcal{C}_S^p$ **to an adjacent cell** X' closer to the target cell C.

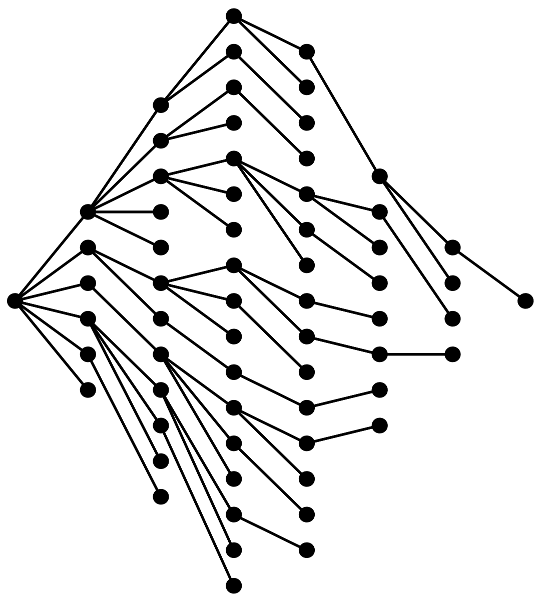




Shoot a ray from an interior point of X to an interior point of C. It will hit all hyperplanes separating X and C. **The first hyperplane hit by the ray is chosen** and its corresponding **index is flipped**, defining $f(X) = X'$.

Once a finite local search f is fixed, we have a uniquely defined **directed spanning tree** T_f of the **cell graph** of $\mathcal{C}_S^p$, **rooted** at C with edge set $\{(X, f(X))|X \in \mathcal{C}_S^p \setminus \{C\}\}$.

Search tree:

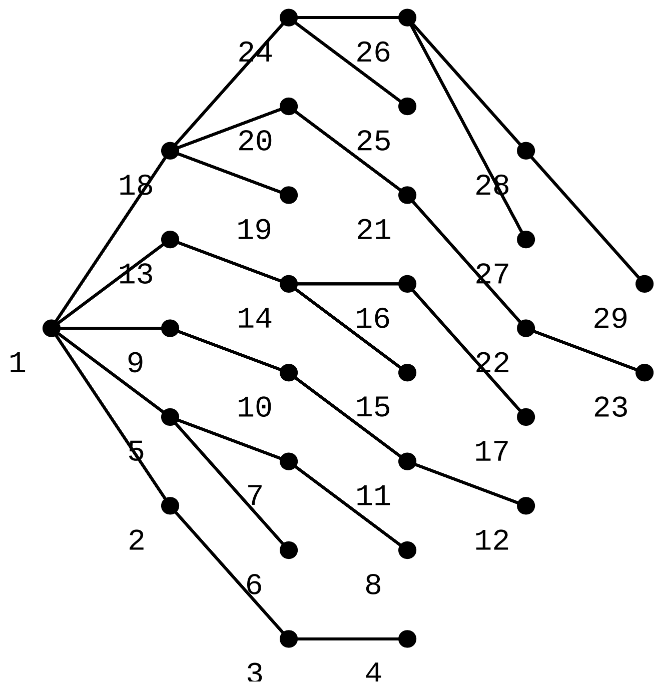


From any cell $X \in \mathcal{C}_S^p$, there is a directed path to C with at most $n-1$ edges giving a maximum height n.

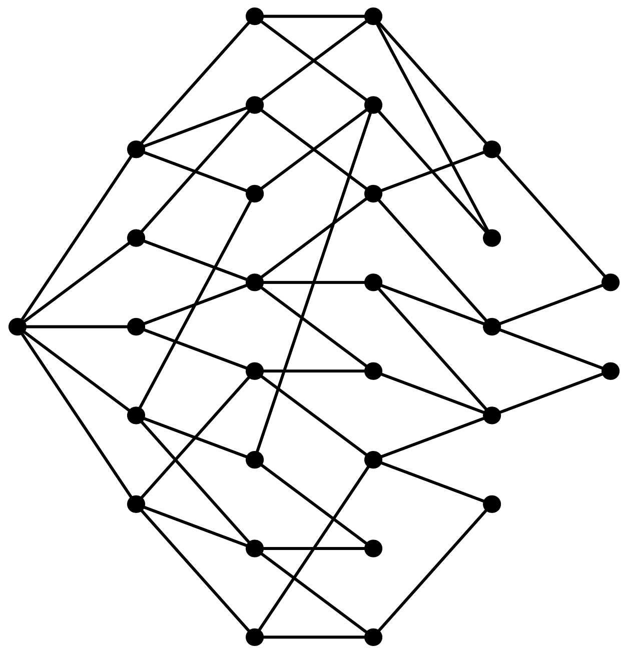
The algorithm is a procedure to **visit all members of** $\mathcal{C}_S^p$ **by tracing the spanning tree** T_f **from** C, relying only on the two functions f and Adj.

Cell enumeration (2 dimension, 7 hyperplanes, 29 cells)

Search Tree:



Cell Graph:



```
procedure BestCellSearch($Adj$,$\delta$,$C$,$f$);
    $X := C$; $depth = |X^+|$; $j := 0$; // $j$: neighbor counter
    **repeat**
        **while** $j < \delta$ **do**
            $j := j + 1$;
            $next := Adj(X, j)$;
            **if** $next \neq 0$ and $g(X) < depth$ **then**
                **if** $f(next) = X$ **then** // reverse traverse
                    $X := next$; $j := 0$;
                    **if** $|X^+| < depth$ **then** $depth = |X^+|$ **endif**
                **endif**
            **endif**
        **endwhile**;
        **if** $X \neq C$ **then** // forward traverse
            $X' := X$;    $X := f(X)$;
            $j := 0$;
            **repeat** $j := j + 1$ **until** $Adj(X, j) = X'$ // restore $j$
        **endif**
    **until** $X = C$ and $j = \delta$;
    output $depth$.
```

Remarks:

The **interior point we select for each cell must be uniquely defined**: $A\,y \leq b$ where A is a $n \times d$ matrix.

$$\begin{array}{llcl} \max & & y_0 & \\ \text{subject to} & A\,y \;+\; e & y_0 & \leq\; b \\ & & y_0 & \leq\; K \end{array}$$

where e is the vector of all 1's and K is any positive number to make the LP bounded. Use a deterministic algorithm, to find a unique solution to this LP.

The **worst case complexity** is $O(n\ \ LP(n,d)\ |\mathcal{C}_S^p|)$ which is the time complexity to generate all cells. To speed up:

1) Let X be the current cell, and Y below X. (X is between Y and C.) An index represent hyperplanes separating Y and C, flipping (crossing) can be done only once. $(X^+ \cap C^-) \subset Y^+$, so $|Y^+| \geq g(X) = |X^+ \cap C^-|$.

Use **branch and bound** technique, with $g(X)$ as bound to obtain **adaptive algorithm**. If $g(X) \geq depth$, the current best depth, stop computing the enumeration of cells below X.

2) **Computation can be improved greatly if C is a good initial cell.** Random choice of a first cell, then local heuristics to improve it locally, checking for a neighbor with smaller number of $+$ positive signs. Then move to this neighbor checking its neighbor etc, until no local improvement is made. Worth to do this **local heuristic search many times parallelly** before starting the deterministic algorithm. TSP uses similar speeding up.

3) **Parallel computation**. At any time the workload of searching below a cell X can be given to another processor. Memory is not shared, only the current best $depth$, which should be updated centrally. **Highly parallelizable**, with m processors the speed up is slighly less than m times, and in higher dimensions it is actually improved.

Tools

1) **cddlib**: **polyhedral computation library** of Fukuda, with implementation of the two basic operations needed, the **linear programming** solving and the **ray shooting** operation.

Since all cddlib functions can be compiled with **both floating-point and rational (exact) arithmetics**, this code can also run in these arithmetics.

2) **ZRAM**: **parallel computation library** of Marzetta, for reverse search, backtracking and branch and bound. ZRAM, takes care of the reverse search mechanism, and provides efficient parallelization at no additional cost.

Let $S = \{p^1, p^2, \dots p^n\} \subset \mathbb{R}^d$ and let h be a hyperplane in $\mathbb{R}^d$. In the dual hyperplane arrangement A_S, there is a point p_h corresponding to h.

The **regression depth** of the hyperplane h is the **minimum number of hyperplanes** of the dual hyperplane arrangement **crossed by any ray starting at the point** p_h.
(Introduced by Rousseeuw and Hubert)

Let $\mathcal{U}$ be the set of all **unbounded cells** in the hyperplane arrangement A_S. The **regression depth of** h:

$$min_{U \in \mathcal{U}} |\sigma(U) - \sigma(p_h)|$$

(There is a signvector $\sigma(p_h)$ corresponding to the point p_h. A direction ϕ corresponds to an unbounded cell U_ϕ. A ray starting at p_h in the direction ϕ will cross as many hyperplanes as there is difference in the signvector of U_ϕ and p_h. The cell enumeration algorithm can be modified to enumerate all unbounded cells, then choose the best.)

Tukey median: a point in $\mathbb{R}^d$ with **max possible depth**

R.Liu is suggesting to **sort the elements of the sample according to their location depth**. This can be done by computing the location depth of each point in the sample S, even parallel, then rank the sample points accordingly.

This can be disappointing as there is possibility that all sample points have data depth 1, if the sample points are in **convex position**.

Much harder is to compute the set of points in $\mathbb{R}^d$ with location depth at least k and the **Tukey median**.

Maximum location depth

For univariate data the **maximum location depth** is $\lfloor \frac{n+1}{2} \rfloor$.

In d-dimension
$\lceil \frac{n}{d+1} \rceil \leq$ **maximum location depth** $\leq \lfloor \frac{n+1}{2} \rfloor$.

Lower bound is obtained for the d-dimensional **simplex**.

Upper bound can be obtained with a **centrally symmetric dataset** and if n odd , with one point in the center.

Convex Geometry gives some theoretical background.

Helly's Theorem: *Suppose K is a family of at least $d+1$ convex sets in $\mathbb{R}^d$, and K is finite or each member of K is compact. Then if each $d+1$ member of K have a common point, then there is a point common to all members of K.*

Let $S \subset \mathbb{R}^d$, $|S| = n \geq (d+1)(k-1)+1$.
$K = \{H | H \text{closed halfspace}, |H \cap S| \geq n - k + 1\}$.
If $H \in K$, then $\overline{H}$ is open halfspace, and $|\overline{H} \cap S| \leq k - 1$.

$|\cup_{i=1}^{d+1} \overline{H_i}| \leq (d+1)(k-1) \Rightarrow |\cap_{i=1}^{d+1} H_i| \geq 1$.
From Helly's theorem it follows that if $k \leq \lceil \frac{n}{d+1} \rceil$
$|\cap K| \geq 1$. Any point in the intersection has location depth at least k.

Proposition: Let S be a pointset of size n in $\mathbb{R}^d$. Let D_k denote the set of points with location depth at least k.

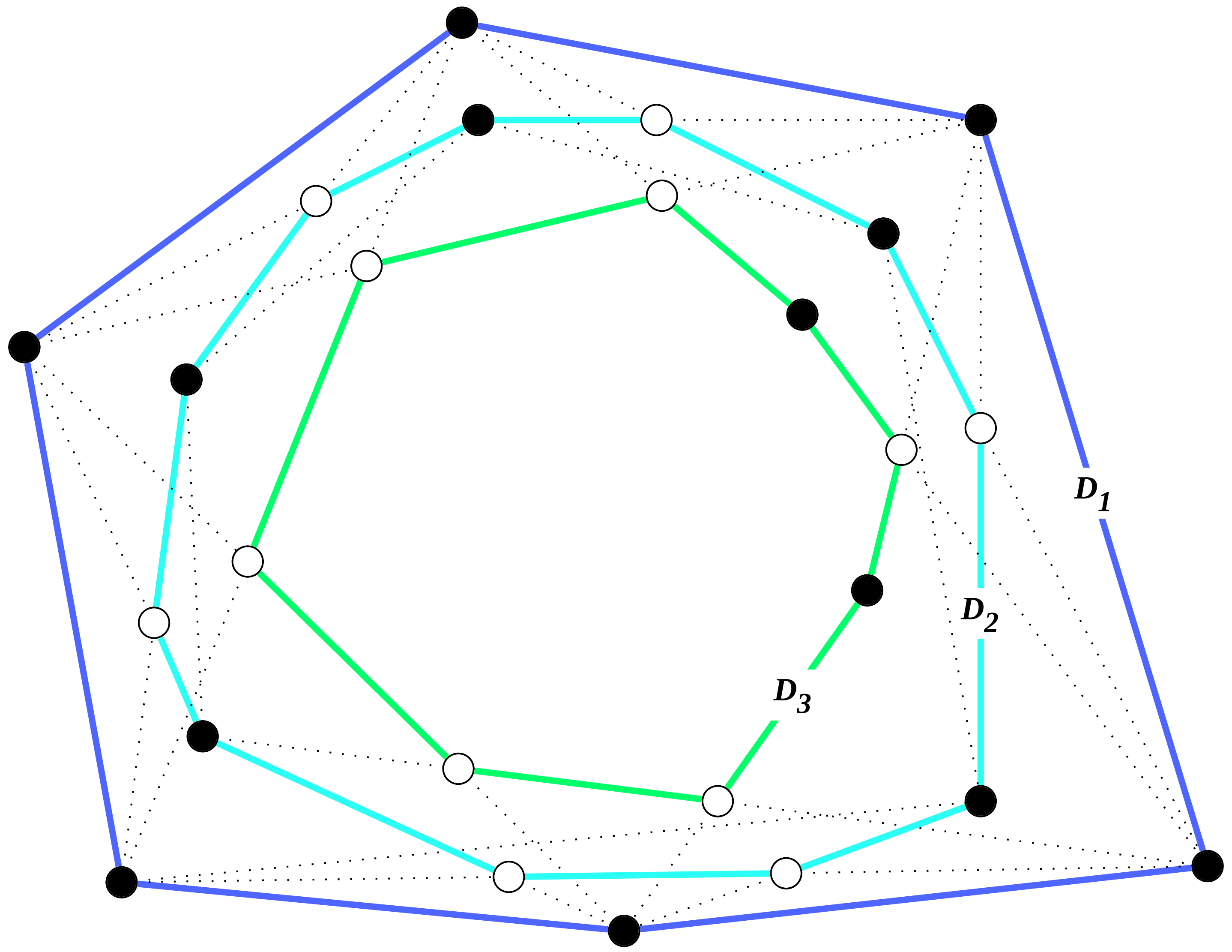
a) D_k is the intersection of all closed halfspaces containing at least $n-k+1$ points of S.

b) $D_{k+1} \subset D_k$, D_k is convex set and D_1 is the convex hull of S.

c) D_k is not empty for all $k \leq \lceil \frac{n}{d+1} \rceil$.

d) D_k is bounded by hyperplanes containing at least d points of S that span a $(d-1)$-dimensional subspace.

$$\lceil \frac{n}{d+1} \rceil \leq \text{maximum of location depth} \leq \lfloor \frac{n+1}{2} \rfloor$$


D_1
D_2
D_3

Previous results:

Location depth region boundary, called also contour, has been computed by an algorithm by Miller et al., in the plane.

$d = 2$: Computing a center point (Cole et al.), computation of center, deepest location, and D_k (Matoušek, Langerman-Steiger, Rousseeuw-Struyf, Miller et al.).

$d = 3$: Computation of center, or just a center point (Naor, Shamir).

Higher dimension:
Clarkson et al. approximation algorithm that finds an **approximate center** in $\mathbb{R}^d$, the iterative algorithm finds with high probability a point with location depths $O(\frac{n}{d^2})$.

Teng extended Johnson and Preparata complexity results: The special case of testing whether a point is a center point is still coNP-complete.

The problem of testing that the Tverberg depth of a point is at least some fixed bound, or of testing whether the point is a Tverberg point is NP-complete.

Amenta et al.: The related problem of testing whether a hyperplane has regression depth at least $n/(d+1)$ is also coNP-complete.

The proposition implies that D_k is a convex polytope if it is not empty, bounded by a finite number of hyperplanes. In order to have an efficient algorithm it would be suitable to find a good characterization of the bounding hyperplanes of D_k as well.

k-th depth region boundary computation

a) Trivial way: Compute all possible candidates for the boundary of $k - th$-regions by enumerating all hyperplanes containing at least d points of S and checking whether one of the closed halfspace contains at least $n - k + 1$ points of S. This computation is $\Theta(n^d d^3)$, even when the boundary is simple.

b) Instead we suggest a more data sensitive method, which hopefully works well for certain data.

Algorithm to compute the location depth contours and the Tukey median

The depth regions computation is closely related to **efficient k-set enumeration**.

In the k-**set problem a hyperplane cleanly separates** k-**points** from the remaining $n - k$ points for an n-point set.

We construct the location depth regions iteratively using the enumeration of all k-sets and their extreme points. The algorithm is adaptive, highly parallelizable and deterministic.

(i) For computing D_{k+1} we **enumerate the k-sets**, these are cells of the dual hyperplane arrangements, with k negative (positive) signs in their signvectors.
(There is already an output sensitive algorithm to compute this.)

(ii) Independently **compute the vertex set of each k-set cells** in the dual arrangement. The vertices with k minuses (plus's) and at least d zero's are the candidates for the boundary of the $k + 1$-th regions in the primal.

(iii) For the boundary we can do **LP redundancy removal**, whose complexity depends only on the actual size n_0, of the boundary, n LP of size $d \times n_0$.

(iv) To decide if D_{k+1} is empty or not one LP is enough, which can be solved easily even in very large scale problem.

All these operations are highly parallelized and depend on the problem at hand.

In the degenerate case it is possible that the boundary of D_{k-1} intersect the boundary of D_k. Then the cells not appearing in the D_{k-1} boundary have exactly k negatives, representing k-sets. If D_{k-1} has been already computed, then to compute the boundary of D_k only the vertices of those cells need to be computed, that have exactly k minuses. This also means that D_{k-1} boundary need to be kept in memory, while D_k is computed.

Remark: Does not seem to exist at present a local characterization for a given hyperplane being facet inducing.

But we think that this is more efficient than the trivial method, as it depends on the complexity of the boundary.

There is little known about this, but it would be reasonable to expect to have simpler boundary as k is increasing.

Convex Geometry

Closely related to Helly's theorem is the

Tverberg theorem. Let d and k be given natural numbers. For any set $A \subset \mathbb{R}^d$ of at least $(d+1)(k-1)+1$ points there exist k pairwise disjoint subsets $A_1, A_2, \ldots, A_k \subset A$ such that the $\cap_{i=1}^{k} \text{conv}(A_i) \neq \emptyset$ (also called k-split).

Any point in the k-split, called Tverberg point, has location depth at least k.

Radon's theorem is a special case with $k = 2$.

k-split $\subset D_k$. (D_k is also called k-core)

$d = 2$: k-split = k-core,

$d > 2$: k-split $\neq$ k-core

(Independent counter examples by Avis, Bárány and Onn already in $\mathbb{R}^3$)