Simplicial Global Optimization

Julius Žilinskas (Юлюс Жилинскас)

Vilnius University, Lithuania

Summer School on Operational Research and Applications
May 16, 2013

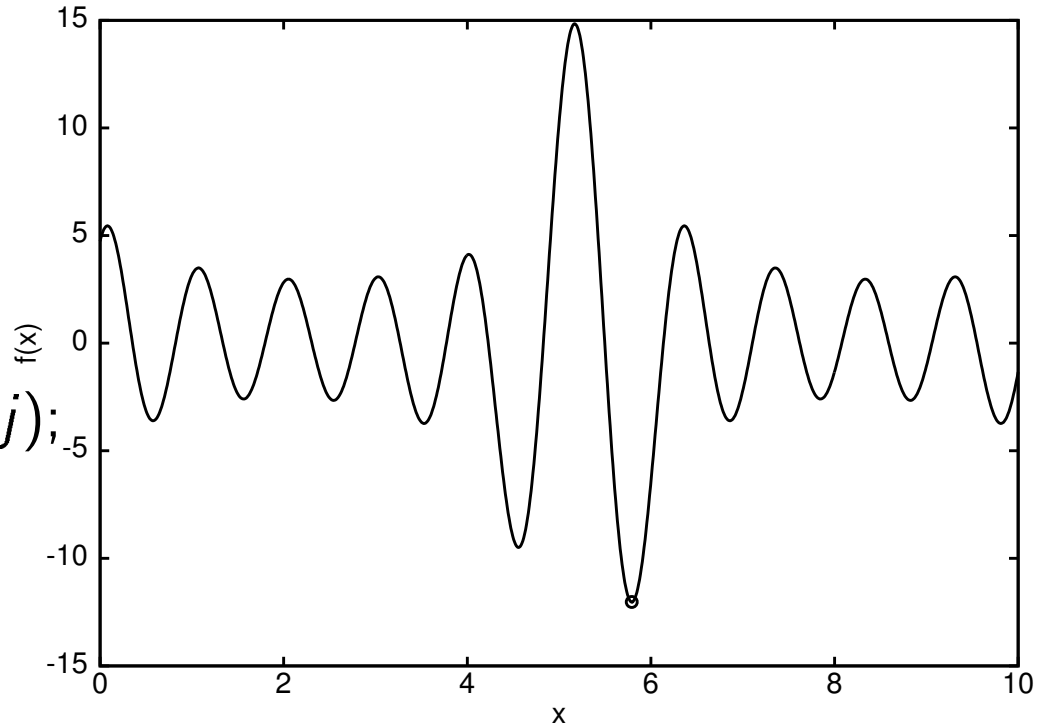
Global optimization

$$\text{Find } f^* = \min_{x \in D} f(x) \text{ and } x^* \in D,\ f(x^*) = f^*, \text{ where } D \subseteq \mathbb{R}^n.$$

Example:

- $n = 1$;
- $D = [0, 10]$;
- Objective function

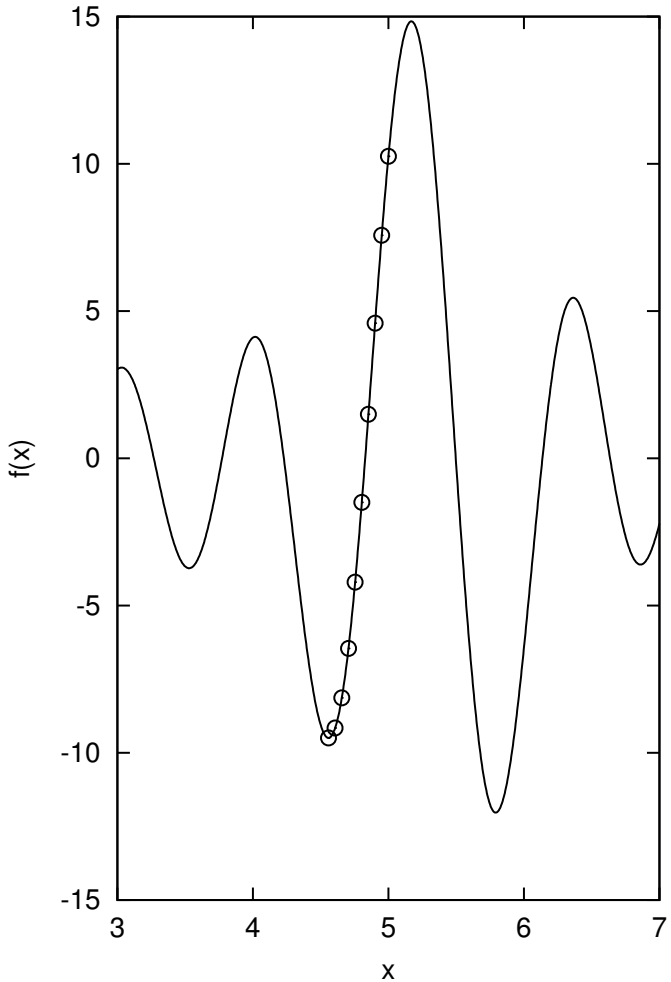
$$f(x) = \sum_{j=1}^{5} j \sin((j+1)x + j);$$



- $f^* = -12.0312$;
- $x^* = 5.7918$.

Local optimization

- A point x^* is a local minimum point if $f(x^*) \leq f(x)$ for $x \in N$, where N is a neighborhood of x^*.
- A local minimum point can be found stepping in the direction of steepest descent.
- Without additional information one cannot say if the local minimum is global.
- How do we know if we are in the deepest hole?



Classification of global optimization methods

- Methods with guaranteed accuracy:
  - Covering methods;
- Direct methods:
  - Random search methods,
  - Clustering methods,
  - Generalized descent methods;
- Indirect methods:
  - Methods approximating level sets,
  - Methods approximating objective function.

Criteria of performance of global optimization algorithms

- Speed:
  - Time of optimization,
  - Number of objective function (and sometimes gradient, bounding, and other functions) evaluations,
  - Both criteria are equivalent when objective function is "expensive" – its evaluation takes much more time than auxiliary computations of an algorithm.
- Best function value found.
- Reliability – how often problems are solved with prescribed accuracy.

Covering methods

- Detect and discard non-promising sub-regions using
  - interval arithmetic $\left\{f\left(X\right)|X \in \underline{X}, \overline{X} \in [\mathbb{R}, \mathbb{R}]^n\right\} \subseteq \overline{f}\left(\overline{X}\right)$,
  - Lipschitz condition $|f(x) - f(y)| \leq L||x - y||$,
  - convex envelopes,
  - statistical estimates,
  - heuristic estimates,
  - ad hoc algorithms.
- May be implemented using branch and bound algorithms.
- May guarantee accuracy for some classes of problems: $f^* \leq \min_{x \in D} f(x) + \epsilon$.

Branch and bound

- An iteration of a classical branch and bound processes a node in the search tree representing a not yet explored subspace.
- Iteration has three main components: selection of the node to process, branching and bound calculation.
- The bound for the objective function over a subset of feasible solutions should be evaluated and compared with the best objective function value found.
- If the evaluated bound is worse than the known function value, the subset cannot contain optimal solutions and the branch describing the subset can be pruned.
- The fundamental aspect of branch and bound is that the earlier the branch is pruned, the smaller the number of complete solutions that will need to be explicitly evaluated.

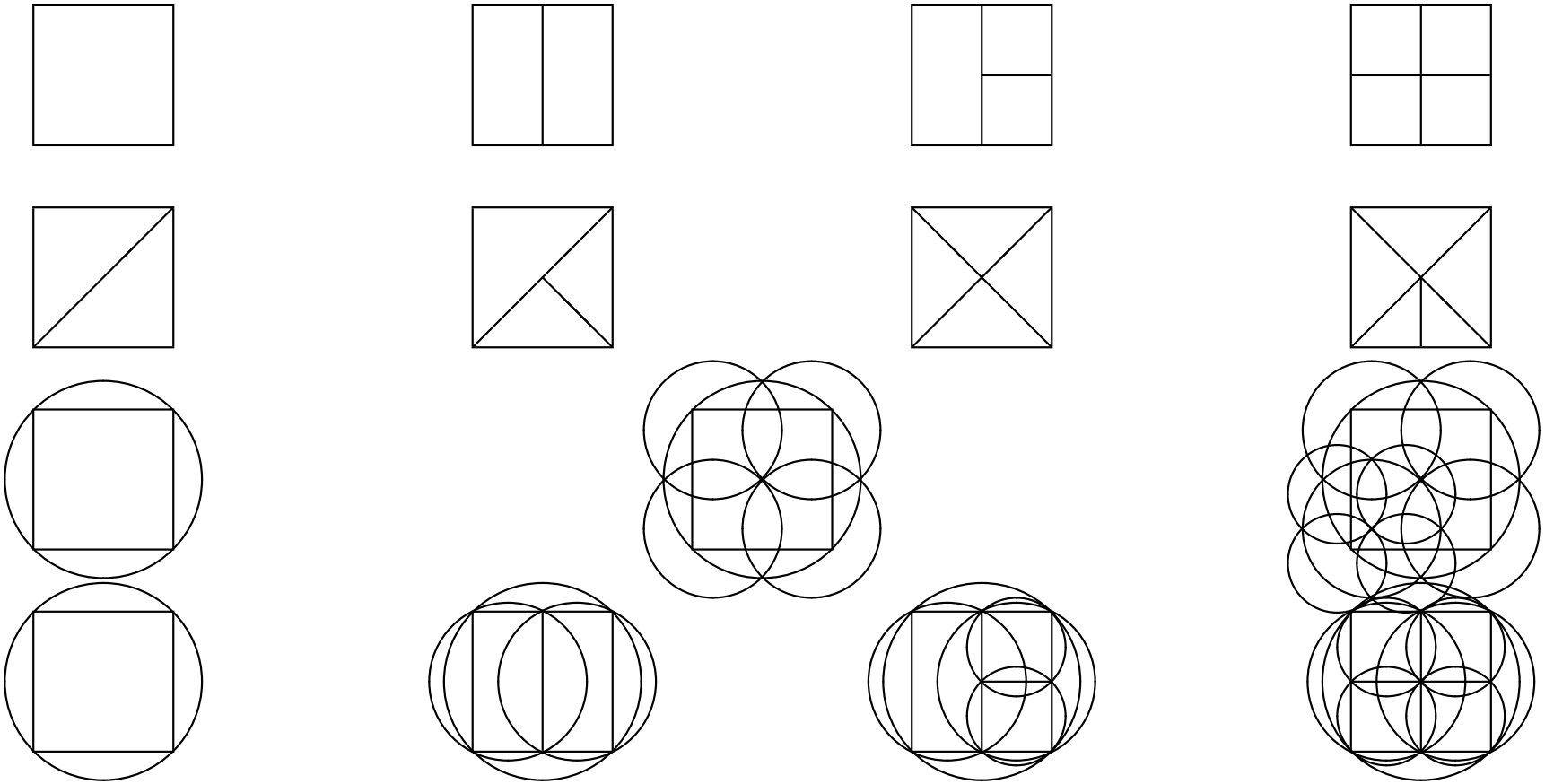
General algorithm for partition and branch-and-bound

Cover D by $L = \{L_j | D \subseteq \bigcup L_j, j = \overline{1, m}\}$ using **covering rule**.
while $L \neq \emptyset$ **do**
  Choose $l \in L$ using **selection rule**, $L \leftarrow L \backslash \{l\}$.
  if bounding rule is satisfied for l **then**
    Partition l into p subsets l_j using **subdivision rule**.
    for $j = 1, \ldots, p$ **do**
      if bounding rule is satisfied for l_j **then**
        $L \leftarrow L \bigcup \{l_j\}$.
      end if
    end for
  end if
end while

Rules of branch and bound algorithms for global optimization

- The **rules** of **covering** and **subdivision** depend on type of partitions used: hyper-rectangular, simplicial, etc.
- **Selection rules**:
  - Best first, statistical – the best candidate: priority queue.
  - Depth first – the youngest candidate: stack or without storing.
  - Breadth first – the oldest candidate: queue.
- **Bounding rule** describes how the bounds for minimum are found. For the upper bound the best currently found function value may be accepted. The lower bound may be estimated using
  - Convex envelopes of the objective function.
  - Lipschitz condition $|f(x) - f(y)| \leq L||x - y||$.
  - Interval arithmetic $\left\{ f(X) \,|X \in \underline{X}, \overline{X} \in [\mathbb{R}, \mathbb{R}]^n \right\} \subseteq \underline{f}\left(\overline{X}\right)$.

Rules of covering rectangular feasible region and branching



Lipschitz optimization

- Lipschitz optimization is one of the most deeply investigated subjects of global optimization. It is based on the assumption that the slope of an objective function is bounded.
- The multivariate function $f : D \to \mathbb{R}, D \subset \mathbb{R}^n$ is said to be Lipschitz if it satisfies the condition

$$|f(x) - f(y)| \leq L\|x - y\|, \quad \forall x, y \in D,$$

  where $L > 0$ is a constant called Lipschitz constant, the domain D is compact and $\|\cdot\|$ denotes a norm.
- Branch and bound algorithm with Lipschitz bounds may be built: if the evaluated bound is worse than the known function value, the sub-region cannot contain optimal solutions and the branch describing it can be pruned.

Lipschitz bounds

The efficiency of the branch and bound technique depends on the bound calculation.

- The upper bound for the minimum is the smallest value of the function at the vertices x_v: $LB(D) = \min_{x_v} f(x_v)$.
- The lower bound for the minimum is evaluated exploiting Lipschitz condition:

$$f(x) \leq f(y) + L\|x - y\|.$$

- The lower bound can be derived as

$$\mu_1 = \max_{x_v \in V(I)} \left\{ f(x_v) - L \max_{x \in I} \|x - x_v\| \right\}.$$

Comparison of selection strategies in Lipschitz optimization

Average number of function evaluations

Dimension	Best First	Statistical	Depth First	Breadth First
$n = 2$	9064	9100	9716	9068
$n = 3$	1217072	1217206	1222060	1217509
$n = 4$	879656	879276	880478	879851
$n = 5, 6$	3939678	3923552	3925984	3960629

Average total amount of simplices

Dimension	Best First	Statistical	Depth First	Breadth First
$n = 2$	18126	18199	19430	18133
$n = 3$	2434138	2434406	244114	2435013
$n = 4$	1759290	1758531	1760934	1759680
$n = 5, 6$	7878996	7846744	7851609	7920898

Average maximal number of simplices in the list

Dimension	Best First	Statistical	Depth First	Breadth First
$n = 2$	2709	640	12	3135
$n = 3$	274960	102734	23	423552
$n = 4$	168927	105814	36	249651
$n = 5, 6$	704862	136589	374	903555

Average execution time

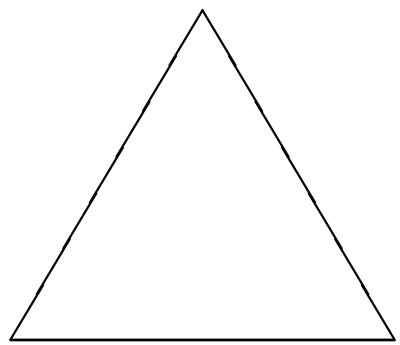
Dimension	Best First	Statistical	Depth First	Breadth First
$n = 2$	0.05	0.05	0.05	0.05
$n = 3$	17.99	16.00	12.41	13.64
$n = 4$	31.65	31.68	26.86	27.53
$n = 5, 6$	1592.19	1568.13	1577.68	1555.55

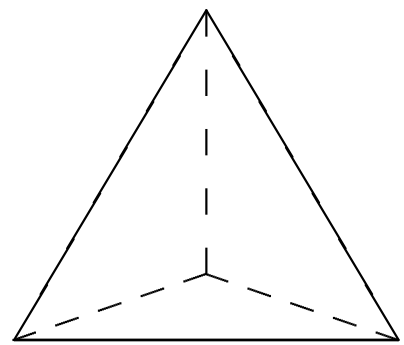
Average ratio $t_{BestFound}/t_{AllTime}$

Dimension	Best First	Statistical	Depth First	Breadth First
$n = 2$	0.234	0.236	0.365	0.408
$n = 3$	0.140	0.008	0.209	0.306
$n = 4$	0.142	0.035	0.240	0.472
$n = 5, 6$	0.074	0.000	0.005	0.077

Simplex

- An n-simplex is the convex hull of a set of $(n+1)$ affinely independent points in n-dimensional Euclidean space.
- An one-simplex is a segment of line, a two-simplex is a triangle, a three-simplex is a tetrahedron.





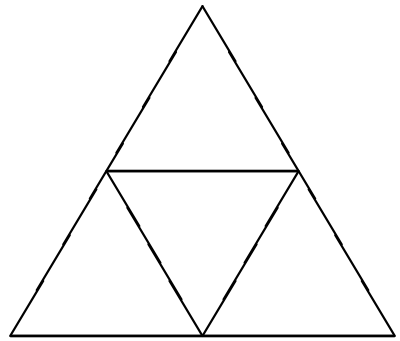
Simplicial vs rectangular partitions

- A simplex is a polyhedron in n-dimensional space, which has the minimal number of vertices.
- Simplicial partitions are preferable when values of the objective function at vertices of partitions are used to evaluate sub-regions.
- Numbers of function evaluations in Lipschitz global optimization with rectangular and simplicial partitions:

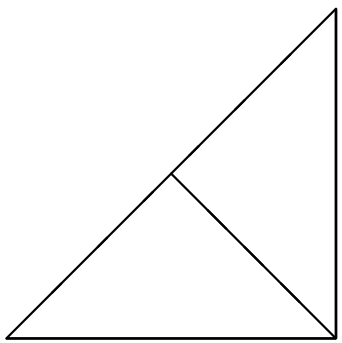
test function	1	2	3	4	5	6	7
rectangular	643	167	3531	45	73	969	7969
simplicial	611	132	2185	70	80	838	3117
test function	8	9	10	11	12	13	
rectangular	301	13953	1123	2677	12643	15695	
simplicial	244	3773	848	1566	4001	4084	

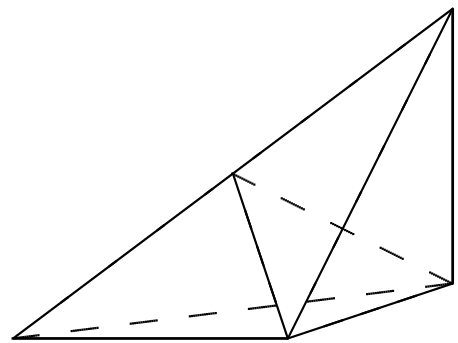
Subdivision of simplices

- Into similar simplices.



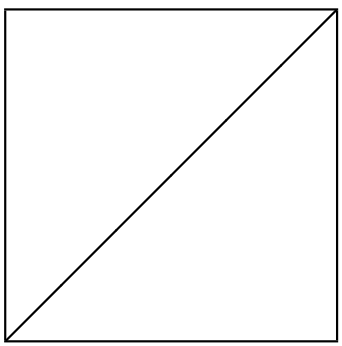
- Into two through the middle of the longest edge.

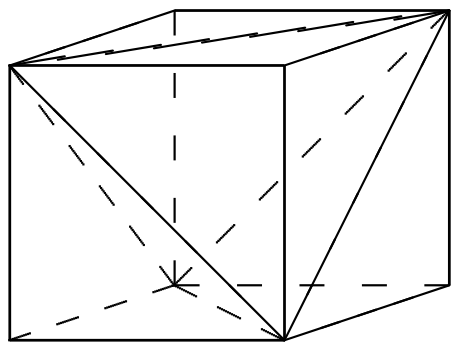


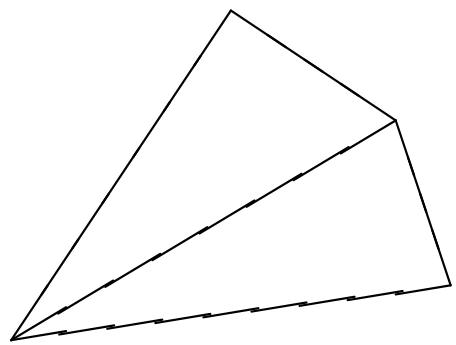


Covering of feasible region

- Very often a feasible region in global optimization is a hyper-rectangle defined by intervals of variables.
- A feasible region is face-to-face vertex triangulated: it is partitioned into n-simplices, where the vertices of simplices are also the vertices of the feasible region.
- A feasible region defined by linear inequality constraints may be vertex triangulated. In this way constraints are managed by initial covering.



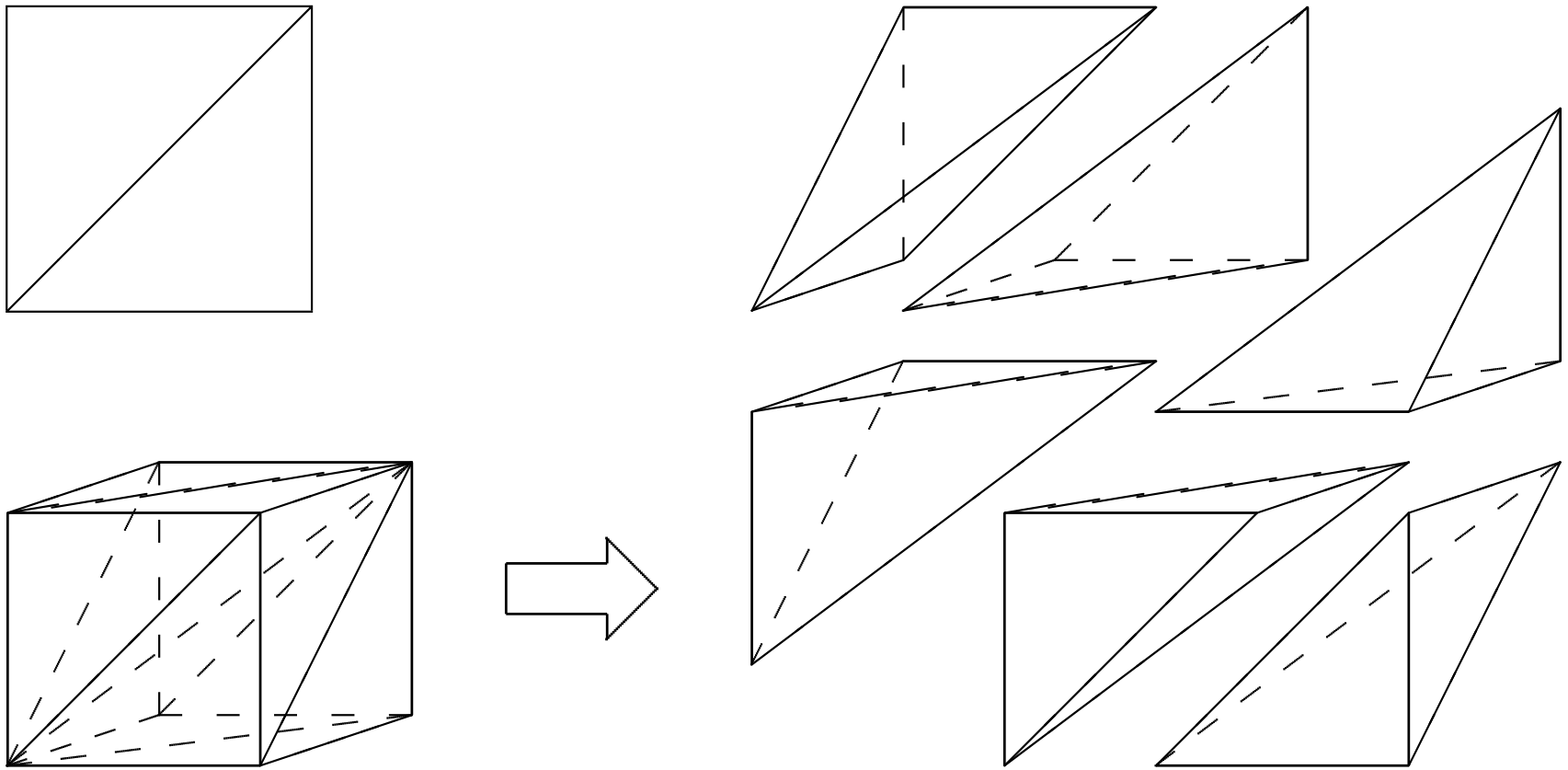




Combinatorial approach for vertex triangulation of a hyper-rectangle

- General – for any n.
- Deterministic, the number of simplices is known in advance – $n!$.
- All simplices are of equal hyper-volume – $1/n!$ of the hyper-volume of the hyper-rectangle.
- By adding just one point at the middle of diagonal of the hyper-rectangle each simplex may be subdivided into two.
- May be easily parallelized.

Examples of combinatorial vertex triangulation of two- and three-dimensional hyper-rectangles



Algorithm for combinatorial vertex triangulation of hyper-rectangle

for $\tau =$ equals one of all permutations of $1, \ldots, n$ **do**
  for $j = 1, \ldots, n$ **do**
    $v_{1j} \leftarrow D_{j1}$
  end for
  for $i = 1, \ldots, n$ **do**
    for $j = 1, \ldots, n$ **do**
      $v_{(i+1)j} \leftarrow v_{ij}$
    end for
    $v_{(i+1)\tau_i} \leftarrow D_{\tau_i 2}$
  end for
end for

Combinatorial vertex triangulation of a unit cube

$$\tau = \{1,2,3\} \qquad v = \begin{Bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{Bmatrix}$$

$$\tau = \{1,3,2\} \qquad v = \begin{Bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{Bmatrix}$$

$$\tau = \{2,1,3\} \qquad v = \begin{Bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{Bmatrix}$$

$$\tau = \{2,3,1\} \qquad v = \begin{Bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{Bmatrix}$$

$$\tau = \{3,1,2\} \qquad v = \begin{Bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{Bmatrix}$$

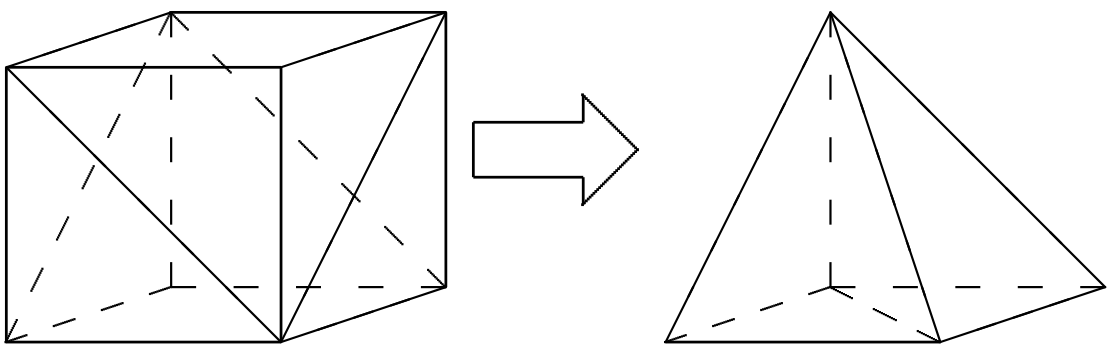
$$\tau = \{3,2,1\} \qquad v = \begin{Bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{Bmatrix}$$

Parallel algorithm for combinatorial vertex triangulation

for $k = \lfloor n!\,rank/size \rfloor$ **to** $\lfloor n!(rank+1)/size \rfloor - 1$ **do**
  for $j = 1, \ldots, n$ **do** $\tau_j \leftarrow j$ **end for**
  $c \leftarrow 1$
  for $j = 2, \ldots, n$ **do**
    $c \leftarrow c(j-1)$
    swap $\tau_{j-\lfloor k/c \rfloor \% j}$ with τ_j
  end for
  for $j = 1, \ldots, n$ **do** $v_{0j} \leftarrow D_{j1}$ **end for**
  for $i = 1, \ldots, n$ **do**
    for $j = 1, \ldots, n$ **do** $v_{(i+1)j} \leftarrow v_{ij}$ **end for**
    $v_{(i+1)\tau_i} \leftarrow D_{\tau_i 2}$
  end for
end for

Vertex triangulation of feasible region defined by linear inequality constraints I

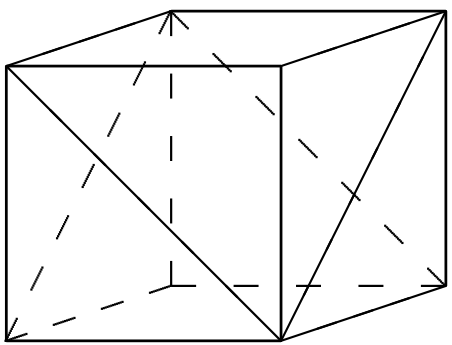
$$\min f(x),$$
$$\text{s.t.} \quad 0 \le x_i \le 1,$$
$$x_1 + x_2 \le 1,$$
$$x_2 - x_3 \le 0.$$

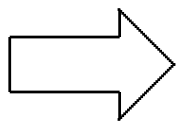


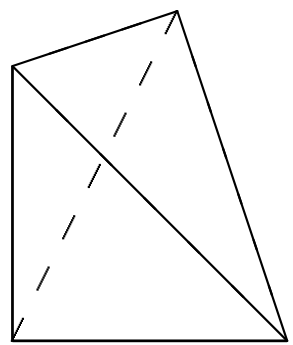
$$\left\{ \begin{array}{l} x_1 = 0, x_2 = 0, x_3 = 0 \\ x_1 = 0, x_2 = 0, x_3 = 1 \\ \cancel{x_1 = 0, x_2 = 1, x_3 = 0} \\ x_1 = 0, x_2 = 1, x_3 = 1 \\ x_1 = 1, x_2 = 0, x_3 = 0 \\ x_1 = 1, x_2 = 0, x_3 = 1 \\ \cancel{x_1 = 1, x_2 = 1, x_3 = 0} \\ \cancel{x_1 = 1, x_2 = 1, x_3 = 1} \end{array} \right\}$$

Vertex triangulation of feasible region defined by linear inequality constraints II

$$\begin{aligned} &\min f(x), \\ &\text{s.t.} \quad 0 \le x_i \le 1, \\ &\qquad\quad x_1 + x_2 \le 1, \\ &\qquad\quad -x_2 + x_3 \le 0. \end{aligned}$$







$$\left\{ \begin{array}{l} x_1 = 0, x_2 = 0, x_3 = 0 \\ \cancel{x_1 = 0, x_2 = 0, x_3 = 1} \\ x_1 = 0, x_2 = 1, x_3 = 0 \\ x_1 = 0, x_2 = 1, x_3 = 1 \\ x_1 = 1, x_2 = 0, x_3 = 0 \\ \cancel{x_1 = 1, x_2 = 0, x_3 = 1} \\ \cancel{x_1 = 1, x_2 = 1, x_3 = 0} \\ \cancel{x_1 = 1, x_2 = 1, x_3 = 1} \end{array} \right\}$$

Coping with symmetries of objective function

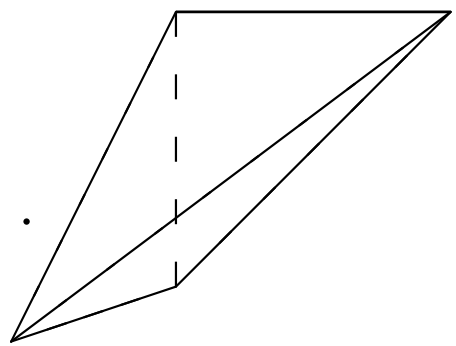
- If interchange of the variables x_i and x_j does not change the value of the objective function, it is symmetric over the hyper-plane $x_i = x_j$.
- It is possible to avoid such symmetries by setting linear constraints on such variables: $x_i \leq x_j$.
- The resulting constrained search space may be vertex triangulated.
- The search space and the numbers of local and global minimizers may be reduced by avoiding symmetries.

Example of coping with symmetries of objective function

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(x_i) + 1, D = [-500, 700]^n$$

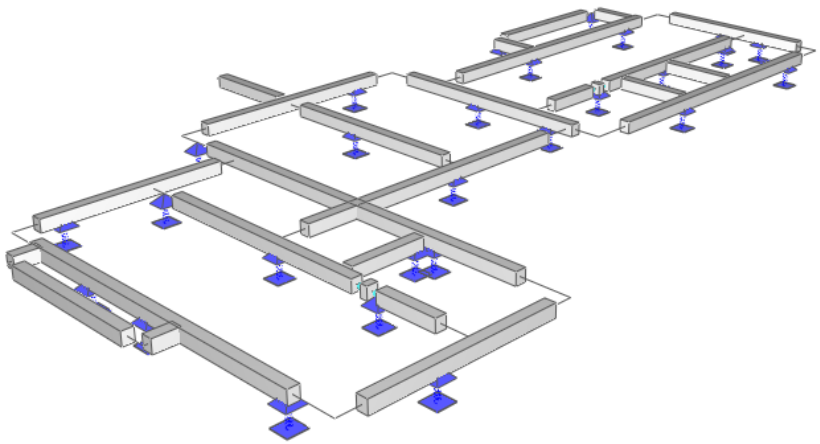
- The objective function is symmetric over hyper-planes $x_i = x_j$.
- Constraints for avoiding symmetries: $x_1 \leq x_2 \leq \ldots \leq x_n$.
- The resulting simplicial search space:

$$D = \left\{ \begin{array}{ccccc} -500 & -500 & \ldots & -500 & -500 \\ -500 & -500 & \ldots & -500 & 700 \\ \vdots & & \ddots & & \vdots \\ -500 & 700 & \ldots & 700 & 700 \\ 700 & 700 & \ldots & 700 & 700 \end{array} \right\}.$$



Example of coping with symmetries: optimization of gillage-type foundations

- Grillage foundation consists of separate beams, which are supported by piles or reside on other beams.
- The piles should be positioned minimizing the largest difference between reactive forces and limit magnitudes of reactions.



Optimization of gillage-type foundations: formulation

- Black-box problem. The values of objective function are evaluated by an independent package which models reactive forces in the grillage using finite element method.
- Gradient may be estimated using sensitivity analysis implemented in the modelling package.
- The number of piles is n.
- The position of a pile is given by a real number, which is mapped to a two-dimensional position by the modelling package. Possible values are from zero to the sum of length of all beams l.
- Feasible region is $[0, l]^n$.
- Characteristic of all piles are equal, their interchange does not change the value of objective function.

Optimization of gillage-type foundations: simplicial search space I

- Let us constrain the problem avoiding symmetries of the objective function: $x_1 \leq x_2 \leq \ldots \leq x_n$.
- Simplicial search space:

$$D = \left\{ \begin{array}{ccccc} 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 0 & l \\ \vdots & & \ddots & & \vdots \\ 0 & l & \ldots & l & l \\ l & l & \ldots & l & l \end{array} \right\}.$$

- Search space and the numbers of local and global minimizers are reduced $n!$ times with respect to the original feasible region.

Optimization of gillage-type foundations: simplicial search space II

- The distance between adjacent piles cannot be too small due to the specific capacities of a pile driver.
- Let us constrain the problem avoiding symmetries and distances between two piles smaller than δ:
  $x_1 \leq x_2 - \delta, \ldots, x_{n-1} \leq x_n - \delta$.
- Simplicial search space:

$$D = \left\{ \begin{array}{cccc} 0 & \delta & \ldots & (n-2)\delta & (n-1)\delta \\ 0 & \delta & \ldots & (n-2)\delta & l \\ \vdots & & \ddots & & \vdots \\ 0 & l-(n-2)\delta & \ldots & l-\delta & l \\ l-(n-1)\delta & l-(n-2)\delta & \ldots & l-\delta & l \end{array} \right\}.$$

Global optimization in least squares nonlinear regression

- The optimization problem in nonlinear LSR:

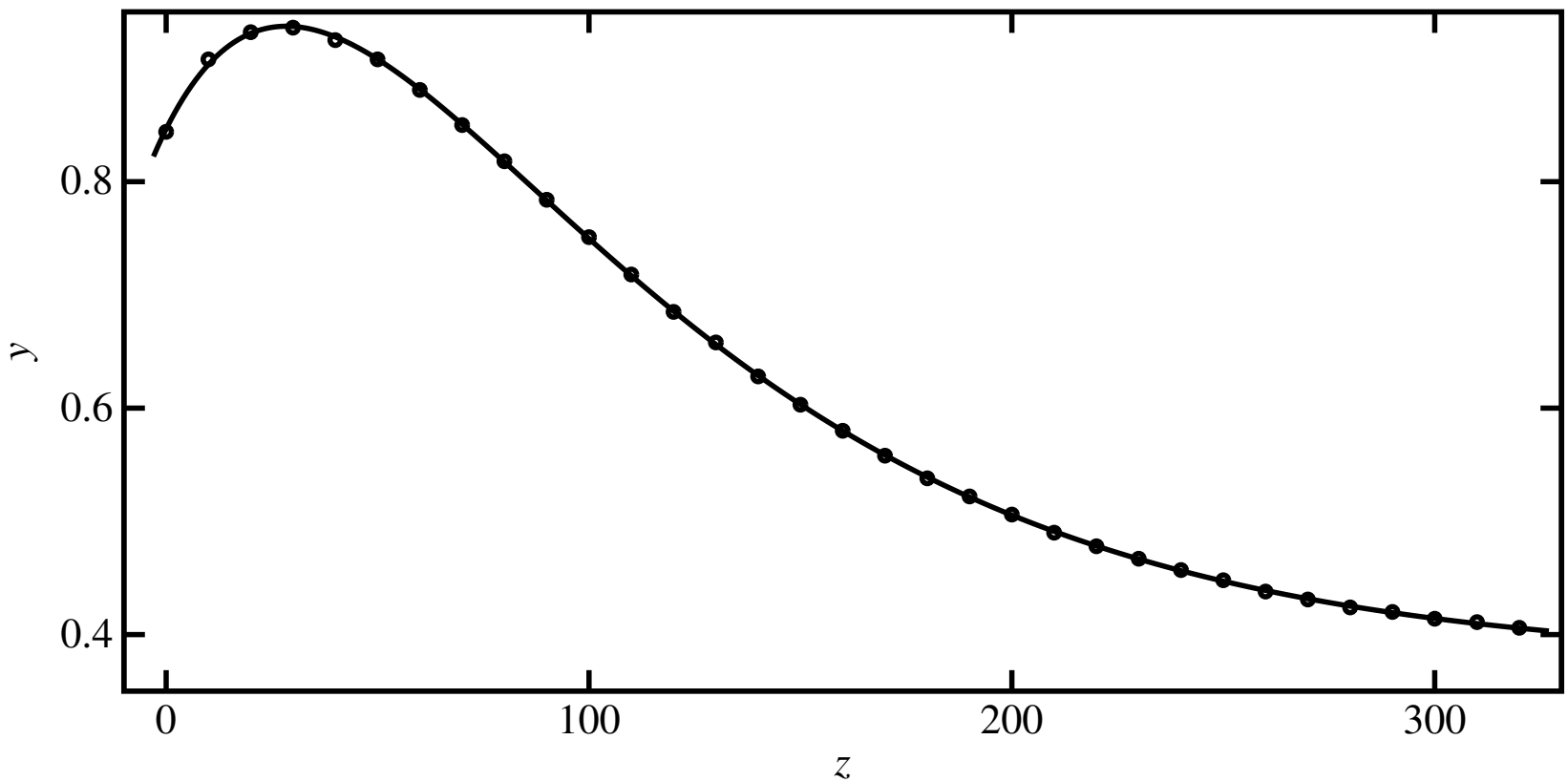
$$\min_{X \in A} \sum_{i=1}^{N} (y_i - \varphi(X, Z_i))^2 = \min_{X \in A} f(X),$$

  where the measurements y_i at the points $Z_i = (z_{1i}, z_{2i}, \ldots, z_{pi})$ should be tuned by the nonlinear function $\varphi(X, \cdot)$.
- The minimization problem seems favorable for application of classical nonlinear programming techniques: normally the number of variables (equal to the number of model parameters to be estimated) is small, and the objective function is smooth.
- A practical problem could be solved easily using well developed nonlinear programming algorithms if a starting point for local descent would be known in the region of attraction of the global minimizer.
- If such a starting point is not known the problem should be considered as a global optimization problem.

Example problem: MGH17

- 1 Response Variable (y)
- 1 Predictor Variable (z)
- 33 Observations

$$\varphi(X, z) = x_1 + x_2 \exp(-x_4 z) + x_3 \exp(-x_5 z)$$



Test problems

No.	Regression function	Feasible region	Minimizer
1	$x_1 x_3 z_1/(1 + x_1 z_1 + x_2 z_2)$	$[0, 10^3]^3$	3.1315, 15.159, 0.7801
2	$x_3(\exp(-x_1 z_1) + \exp(-x_2 z_2))$	$[0, 10^3]^2$, $[0, 10^4]$	13.241, 1.5007, 20.100
3	$x_3(\exp(-x_1 z_1) + \exp(-x_2 z_2))$	$[0, 2 \cdot 10^3]$, $[0, 10^2]^2$	814.97, 1.5076, 19.920
4	$x_1 + x_2 \exp(x_3 z)$	$[0, 10^3]$, $[0, 10]$, $[0, 2]$	15.673, 0.9994, 0.02222
5	$x_1 \exp(x_2/(x_3 + z))$	$[0, 10^4]$, $[0, 5 \cdot 10^4]$, $[0, 10^4]$	$5.610 \cdot 10^{-3}$, 6181.4, 345.22
6	$\exp(x_1 z) + \exp(x_2 z)$	$[0, 1]$, $[0, 1]$	0.2578, 0.2578
7	$x_1 \exp(x_3 z) + x_2 \exp(x_4 z)$	$[0, 10^8]^2$, $[-2, 0]$, $[-5, 0]$	1655.2, $3404 \cdot 10^4$, -0.6740, -1.8160
8	$x_1 z^{x_3} + x_2 z^{x_4}$	$[0, 1]$, $[1, 8]$, $[1, 5]$, $[0, 1]$	0.004141, 3.8018, 2.0609, 0.2229
9	$\exp(x_1 z) + \exp(x_2 z)$	$[-50, 50]^2$	0.2807, 0.4064
10	$x_1 + x_2 \exp((x_3 + x_4 z)^{x_5})$	$[0, 100]^2$, $[0, 5]$, $[0, 1]^2$	9.3593, 2.0292, 1.3366, 0.4108, 0.3551
11	$x_1 \exp(x_3 z) + x_2 \exp(x_4 z)$	$[-10^4, 10^4]^2$, $[-1, 1]^2$	47.971, 102.05, -0.2466, -0.4965
12	$x_1 z^{x_2} + x_3^{x_2/z}$	$[0, 5]^3$	0.05589, 3.5489, 1.4822
13	$x_1 + x_2 z_1^{x_3} + x_4 z_2^{x_5} + x_6 z_3^{x_7}$	$[-5, 5]$, $\{[-5, 5], [0, 10]\}^3$	1.930, 2.578, 0.8017, -1.299, 0.8991, 0.01915, 3.018
14	$x_1 \ln(x_2 + x_3 z)$	$[0, 100]^3$	2.0484, 18.601, 1.8021

Application of CToolbox interval algorithm

No.	Time (s)	NFE	Minimum
1	3600	718,757	Not finished
2	3600	1,723,972	Not finished
3	3600	1,186,617	Not finished
4	3600	1,191,193	Not finished
5	3600	789,384	Not finished
6	0.00	661	124.3622
7	3600	656,474	Not finished
8	3600	672,142	Not finished
9	0.54	13,570	$8.896301 \cdot 10^{-3}$
10	3600	555,223	Not finished
11	3600	700,494	Not finished
12	1.29	17,454	$4.375281 \cdot 10^{-3}$
13	3600	659,215	Not finished
14	3600	1,644,024	Not finished

Modified problems

- When the regression function contains linear and nonlinear parameters the optimal values of the former can be found analytically reducing the dimensionality of the problem.
- The regression function $\varphi(X, z) = x_1 \exp(x_3 z) + x_2 \exp(x_4 z)$ contains the linear parameters x_1 and x_2 whose optimal values can be expressed as

$$b_1 = \frac{(\sum y_i \exp(x_3 z_i)) \times (\sum \exp(2x_4 z_i)) - (\sum y_i \exp(x_4 z_i)) \times (\sum \exp((x_3 + x_4) z_i))}{(\sum \exp(2x_3 z_i)) \times (\sum \exp(2x_4 z_i) - (\sum \exp((x_3 + x_4) z_i))^2}$$

$$b_2 = \frac{(\sum y_i \exp(x_4 z_i)) \times (\sum \exp(2x_3 z_i)) - (\sum y_i \exp(x_3 z_i)) \times (\sum \exp((x_3 + x_4) z_i))}{(\sum \exp(2x_3 z_i)) \times (\sum \exp(2x_4 z_i)) - (\sum \exp((x_3 + x_4) z_i))^2}$$

with the subsequent reduction of the original four-dimensional problem to a two-dimensional problem:

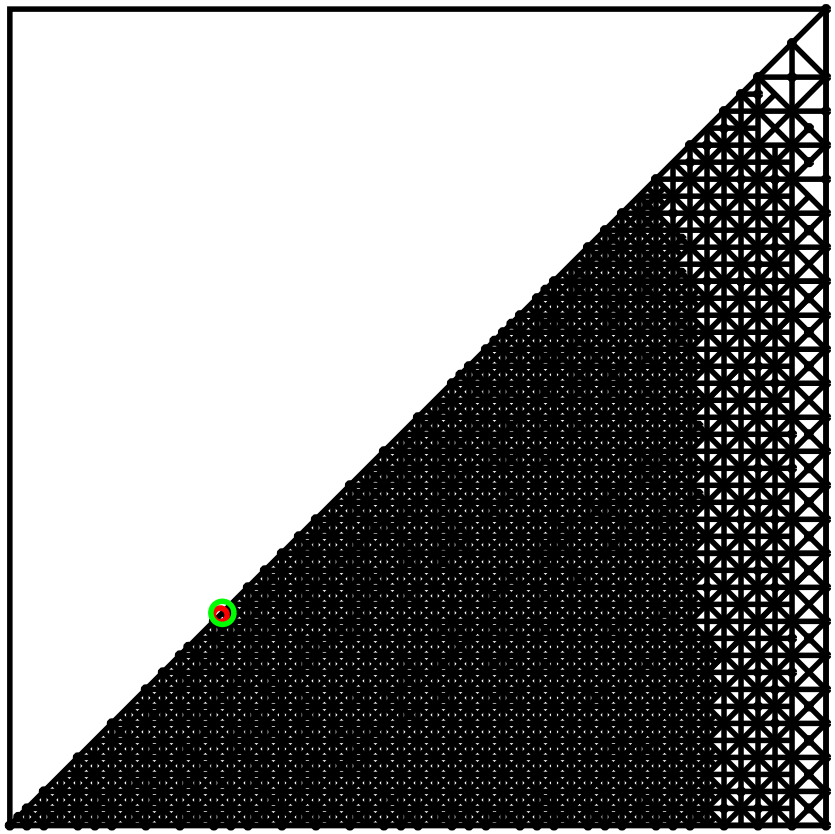
$$\varphi(X, z) = b_1 \exp(x_3 z) + b_2 \exp(x_4 z).$$

Optimization of modified test problems using CToolbox

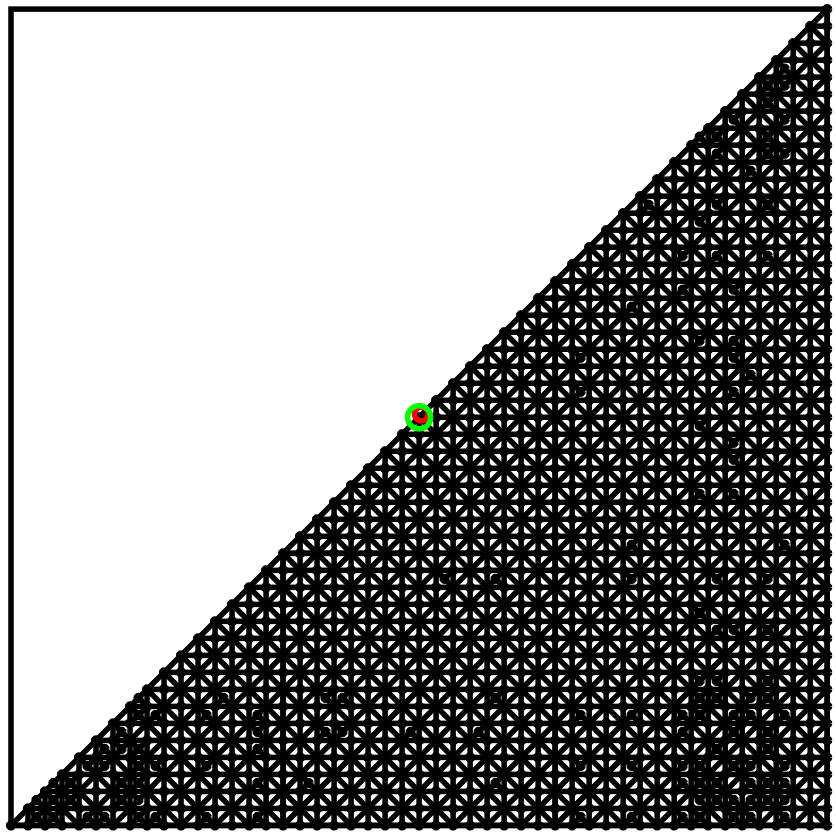
Test No.	Time	NFE	Minimum
1	0.58	11,572	$4.355266 \cdot 10^{-5}$
2	173.49	398,660	$7.471221 \cdot 10^{-5}$
3	3600	1,054,775	not finished
12	0.68	6,507	$4.375281 \cdot 10^{-3}$
4	error of division by interval containing 0		
5	error inverting matrix in interval Newton method		
6-14	error of division by interval containing 0		

Optimization with simplicial Lipschitz branch and bound with global constant

Test problem 6

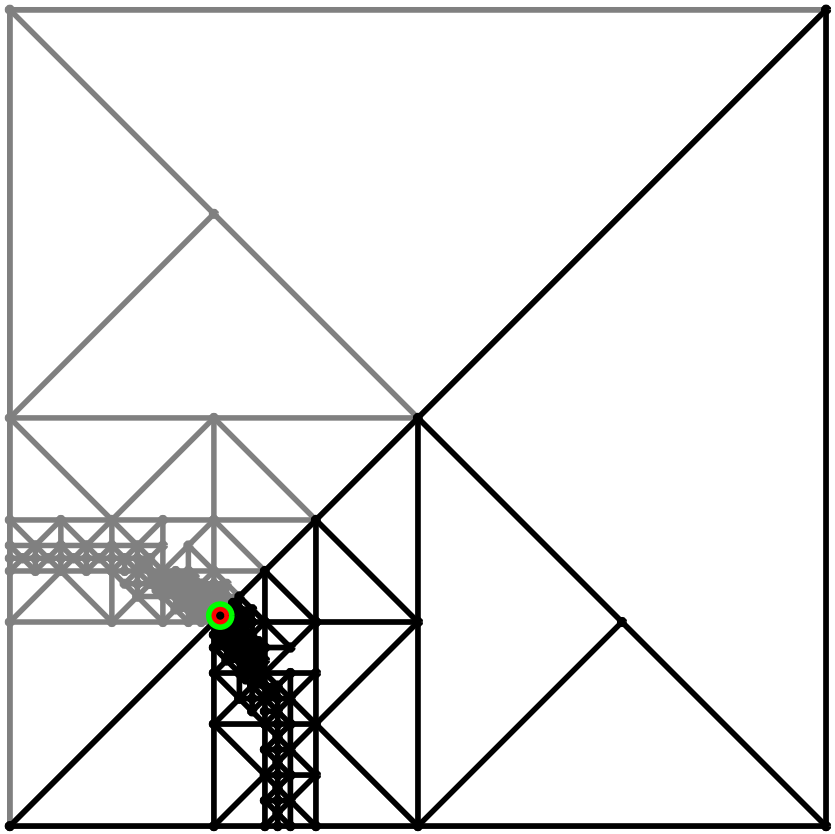


Test problem 9

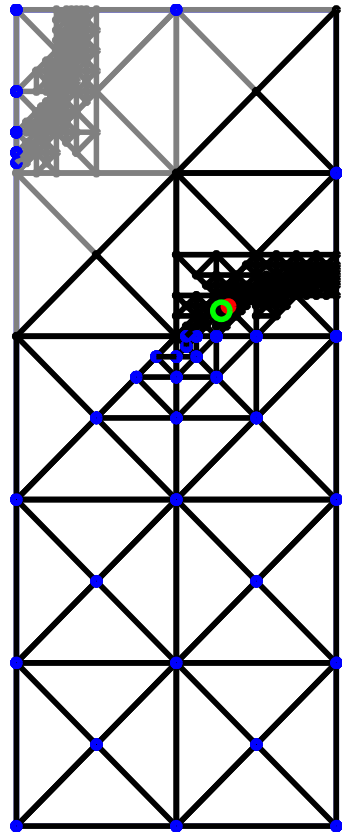


Optimization with simplicial algorithm

Test problem 6

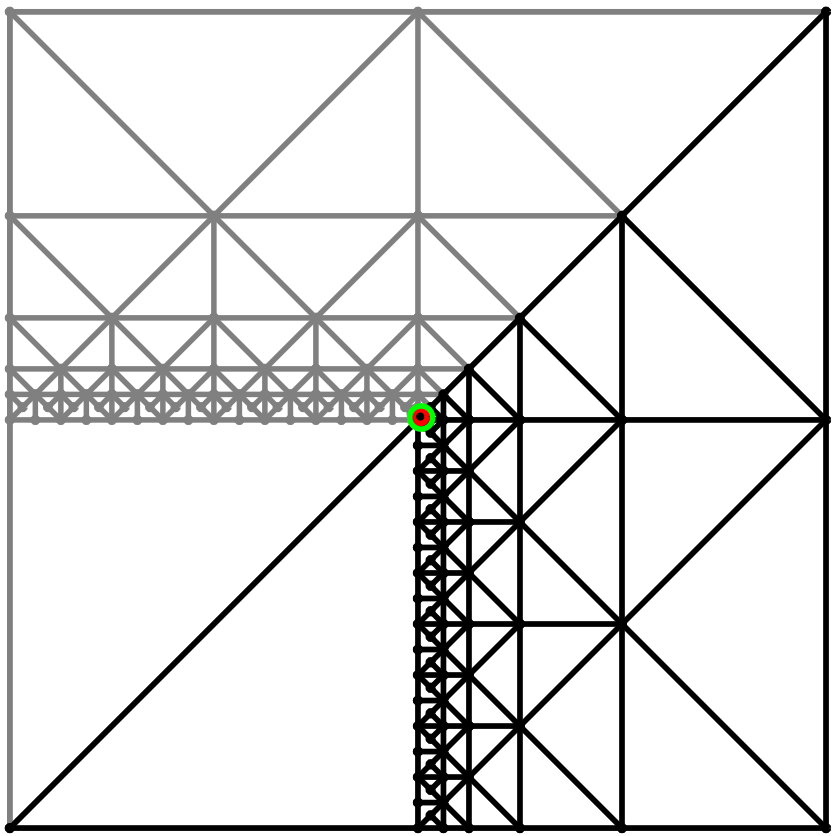


Test problem 7

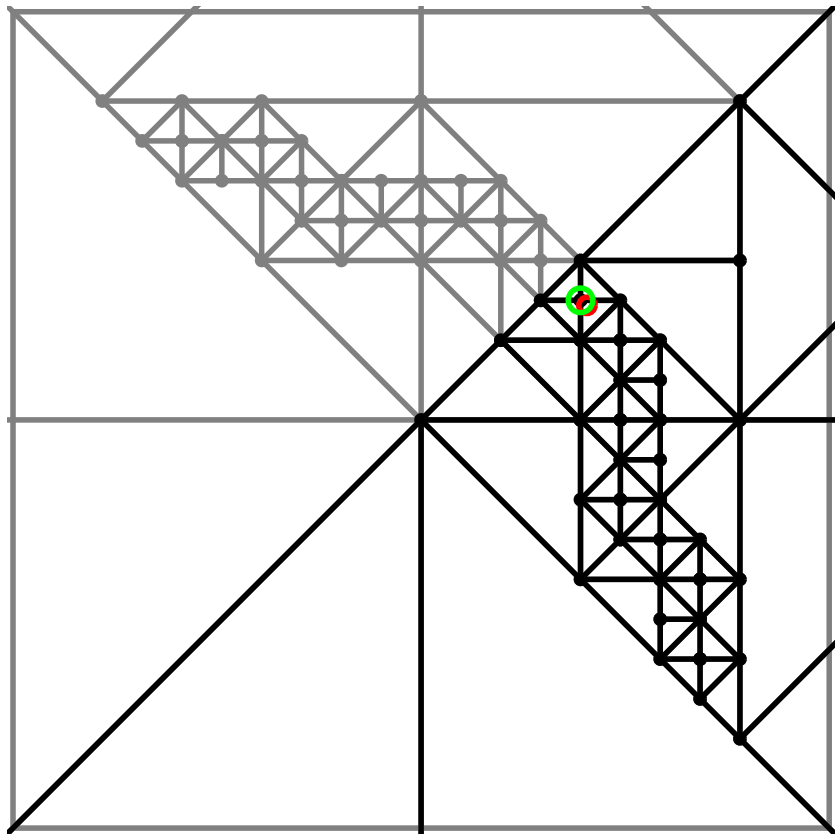


Optimization with simplicial algorithm

Test problem 9

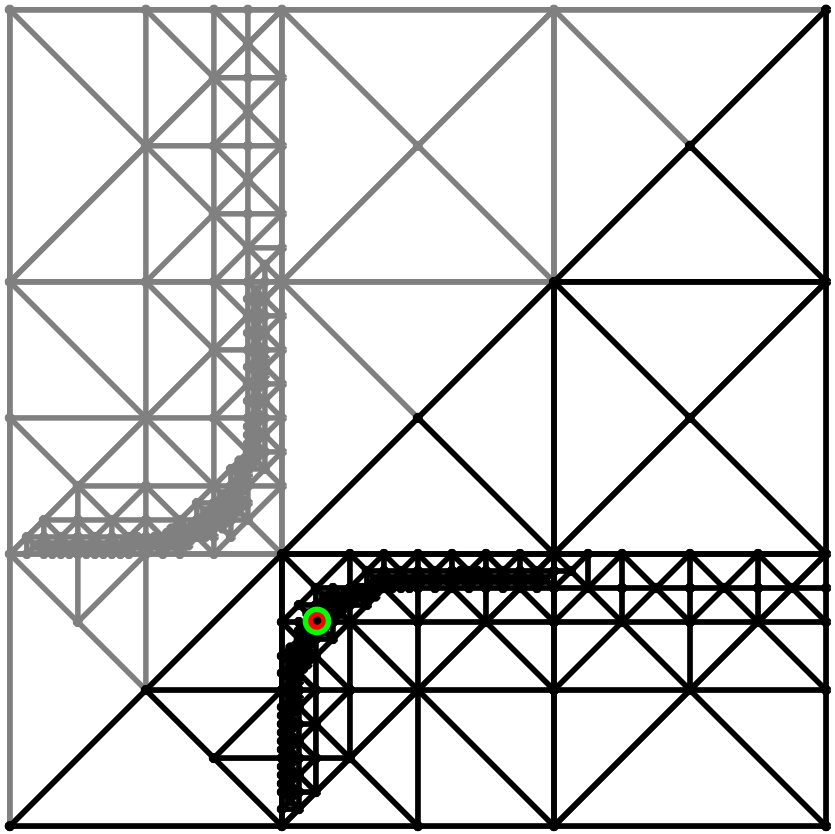


zoomed in

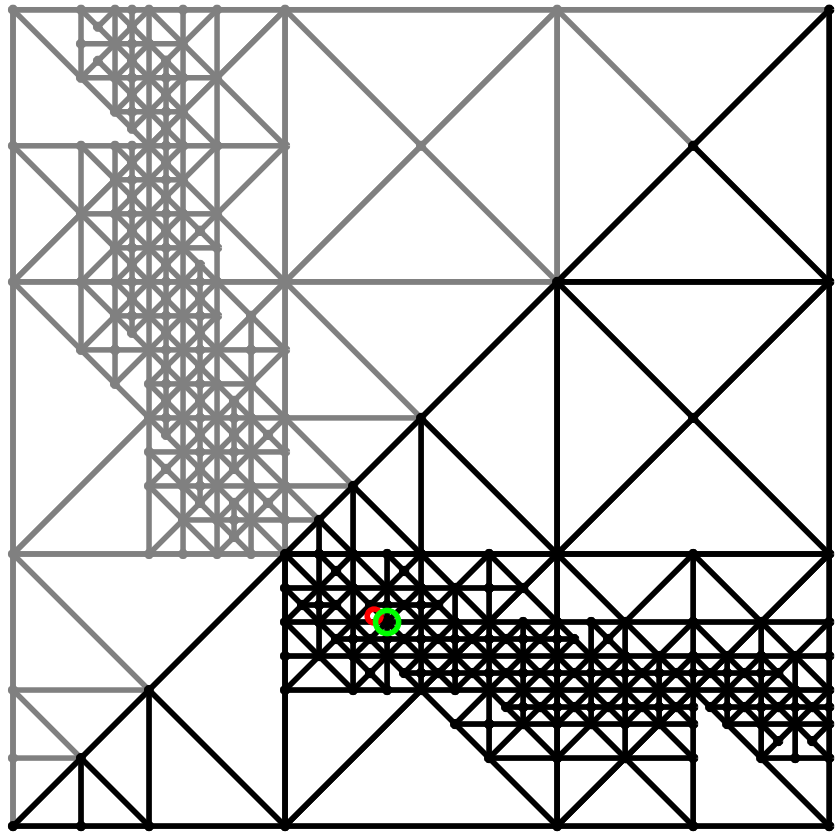


Optimization with simplicial algorithm

Test problem 11



Problem MGH17



Optimization with simplicial algorithm

Problem	n	n'	NFE	NFE avoiding symetries
Test 6	2	2	600	302
Test 7	4	2	574	306
Test 9	2	2	317	161
Test 11	4	2	2101	1089
MGH17	5	2	573	290
Lanczos1	6	3	42692	7118
Lanczos2	6	3	42692	7118
Lanczos3	6	3	42758	7129

Heuristic attraction based subdivision method

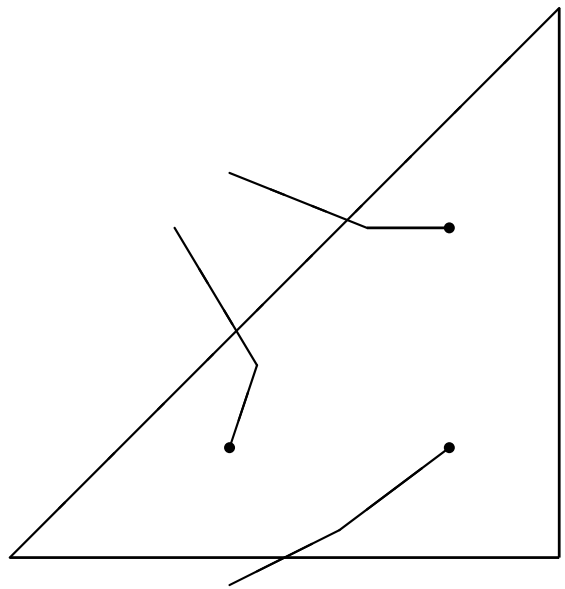
- Developed for multidimensional global optimization.
- Partitioning is controlled by the information acquired during local searches.
- There is no guarantee that the global minimum is found with a prescribed accuracy.
- Applicable in a “black box” situation.
- Algorithm with hyper-rectangular partitions has been applied for multidimensional scaling, many-body problems, growth model of human mandible problem, grillage-type foundation problem.

Attraction based subdivision algorithm

```
while not time-limit do
        while sub-region list is not empty do
                remove best sub-region
                apply local searches from the sample points
                reject or subdivide
        end
        for all rejected sub-regions do
                subdivide and add to sub-region list
        end
end
```

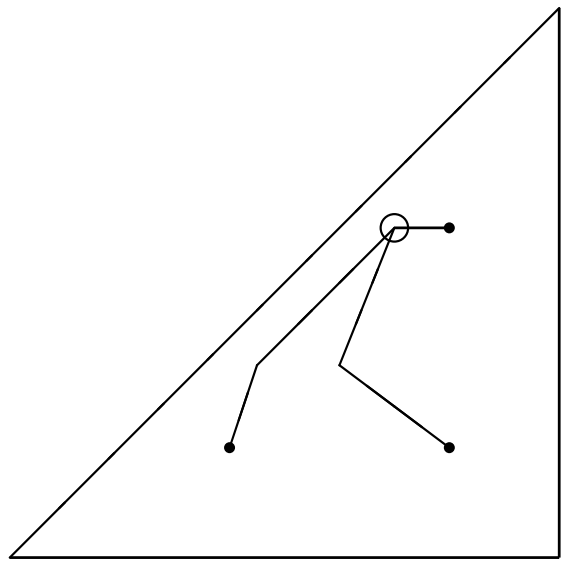
Rejection of the sub-region I

- All local searches go outside of the sub-region.
- Say, there is no minimum point in the sub-region.



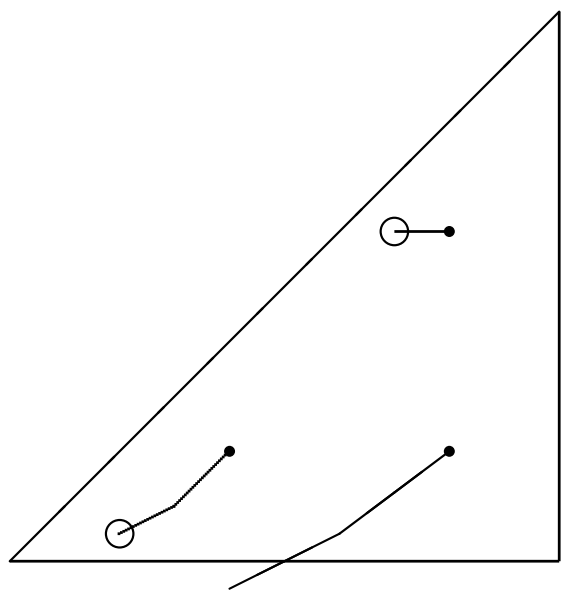
Rejection of the sub-region II

- All local searches converge to the same point.
- Say, there is only one minimum point in the sub-region.



Rejection of the sub-region III

- The estimated lower bound for the minimum over the sub-region is larger than the smallest function value found before. The lower bound is estimated using the largest gradient norm and the smallest objective function value in the sub-region obtained during local searches.



Algorithmic copositivity detection by simplicial partitioning

- A symmetric matrix A is called *copositive* if $x^T Ax \geq 0$ for all $x \in \mathbb{R}^n_+$, where $\mathbb{R}^n_+ := \{x \in \mathbb{R}^n : x_i \geq 0 \text{ for all } i\}$ denotes the non-negative orthant.
- Bundfuss and Dür (2008) proposed algorithmic copositivity detection by simplicial partition.
- The algorithm starts with a standard simplex, whose vertices are the unit vectors $e_1, \ldots, e_n$.
- Simplices are subdivided until:
  - $v^T Av < 0$ for one vertex v of one of the simplices what means that the matrix A is not copositive *or*
  - $v_i^T Av_j \geq 0$ for all pairs of vertices v_i and v_j of all simplices what means that the matrix A is copositive.
- Depth first selection strategy without storing the set of simplices and corresponding matrices may be applied to avoid memory problems.

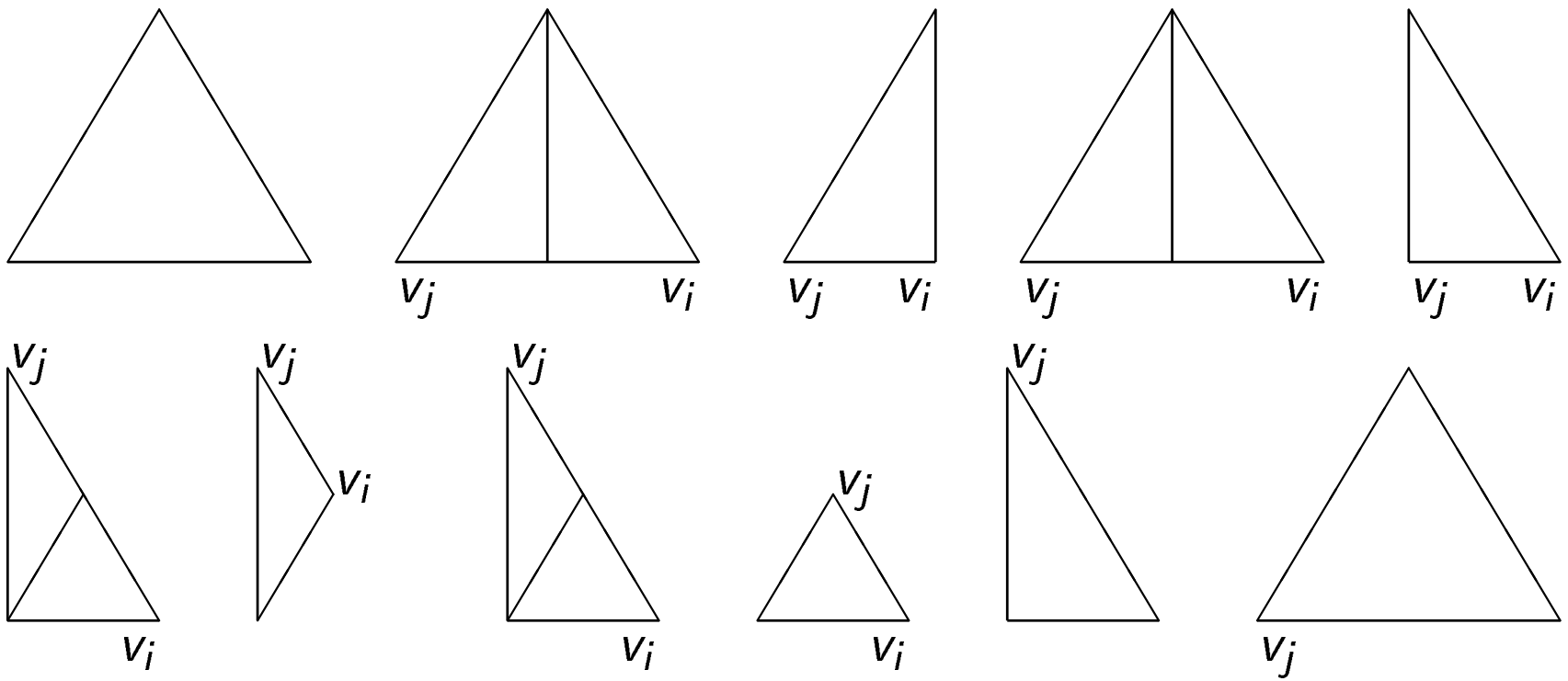
Sketch of the algorithm for copositivity detection

Start with the standard simplex
while not stopped **do**
  if $v^T Av < 0$ for at least one vertex v of the simplex **then**
    the matrix is not copositive, exit
  else if $v_i^T Av_j \geq 0$ for all vertices v_i and v_j of the simplex **then**
    restore vertices, change vertex or stop the cycle
    remember the changed vertex
  else
    subdivide the edge with the smallest $v_i^T Av_j$
    remember the changed vertex
  end if
end while
the matrix is copositive

Notes on the algorithm for copositivity detection

- We use a matrix Q to store the values $v_i^T A v_j$ so not to compute them for each simplex.
- A matrix V is used to store the vertices of a simplex.
- We start with $V \leftarrow (e_1, \ldots, e_n)$; $Q \leftarrow A$.
- Storing the matrices V and Q of *all* candidate simplices would require a large amount of memory.
- We apply a depth-first selection strategy without storing the whole set of simplices, thus we store only the information required to restore V and Q when returning to the lower level (l).
- i_l^* and j_l^* are the row and column numbers of the smallest negative element of Q and the numbers of vertices which are used in subdivision of simplex.
- A vector v_l^* is used to restore the vertex of the simplex, whereas a vector q_l^* is used to restore row and column of the matrix Q.

Subdivision, remembering, restoring



Experimental results on random matrices

$A_{ij} = A_{ji} = rand(-1, 1),\ A_{ii} = 1,\ i, j = 1, \ldots, n.$

	copositive						$\overline{t}*$, ms
n	#	%	max n_{simp}	max l	$\overline{t}$, μs	$\overline{t}_{cop}/\overline{t}_{ncop}$	(Yang&Li'09)
3	904408	90.4408	47	12	8.58	1.8	0.1
4	707647	70.7647	381	19	15.79	2.5	0.7
5	465611	46.5611	3125	25	27.97	4.5	3.2
6	252559	25.2559	7231	28	45.95	9.0	19.1
7	112081	11.2081	53777	31	66.86	19.0	96.4
8	39285	3.9285	55091	36	83.17	45.1	398.7
9	10791	1.0791	112423	35	91.59	119.4	351.2
10	2191	0.2191	199551	34	81.92	372.2	363.5
11	294	0.0294	238983	30	68.34	1599.3	
12	37	0.0037	1283566	38	63.36	3562.0	
13	6	0.0006	491129	27	46.63	14464.7	
14	0	0.0000	69329	29	53.70		
15	0	0.0000	38487	34	67.08		
16	0	0.0000	124	27	75.94		
17	0	0.0000	21	20	96.33		
18	0	0.0000	18	17	113.38		
19	0	0.0000	19	18	138.80		
20	0	0.0000	19	18	157.93		

Experimental results on random copositive matrices

$$A = BB^T + N,\ B_{ij} = rand(-1,1),\ N_{ij} = rand(0,1),\ i,j = 1,\ldots,n.$$

	possitive semidefinite		purely non-negative				
n	#	%	#	%	max n_{simp}	max l	$\bar{t}$, ms
3	6721	67.21	5288	52.88	15	4	0.006
4	5339	53.39	2407	24.07	39	5	0.009
5	4215	42.15	799	7.99	83	8	0.017
6	3299	32.99	183	1.83	249	11	0.036
7	2541	25.41	40	0.40	495	12	0.087
8	1891	18.91	10	0.10	1475	15	0.242
9	1453	14.53	1	0.01	10701	21	0.973
10	1045	10.45	0	0.00	20465	22	2.572
11	799	7.99	0	0.00	40377	23	9.524
12	550	5.50	0	0.00	285739	26	35.72
13	403	4.03	0	0.00	1627925	29	147.8
14	255	2.55	0	0.00	3764717	34	621.5
15	179	1.79	0	0.00	7614199	36	2502.8
16	120	1.20	0	0.00	75535303	39	11129.8
17	72	0.72	0	0.00	307585243	43	57932.7

Copositivity detection for solution of copositive programs

- A quadratic programming problem with a single quadratic constraint

$$\min\langle Q, X\rangle \ \text{ s.t. } \langle D, X\rangle = b,\ X = xx^T,\ x \geq 0.$$

- Copositive formulation with a variable $y \in \mathbb{R}$ and the cone of copositive matrices $\mathcal{C}$

$$\max\{y :\ Q - yD \in \mathcal{C}\}.$$

- The maximum clique problem may be formulated as

$$\omega(G) = \min\{t :\ tQ - J \in \mathcal{C}\},$$

where $t \in \mathbb{N}$ is a variable, $\omega(G)$ is the clique number, and $Q = J - A_G$, where A_G is the adjacency matrix of the graph G, J is a matrix with all entries equal to 1.

- Since there is a single variable (y or t), the problems may be solved by performing a series of copositivity tests.

Results on matrices for generated MaxClique problems

Graph	n	m	ω	A	$A \in \mathcal{C}$?	n_{simp}	max l	t, s
Brock20	20	95	5	$4Q - J$	No	7	6	0
				$5Q - J$	Yes	1599352423	58	24297
Jagota16	16	57	8	$6Q - J$	No	13	12	0
				$7Q - J$	No	20968165	56	201
				$8Q - J$	Yes	130533619	63	1254
Morgen20	20	67	5	$4Q - J$	No	9	8	0
				$5Q - J$	Yes	1263040427	49	21621
Morgen22	22	68	5	$4Q - J$	No	11	10	0
				$5Q - J$	Yes	110514789	47	2225
Sanchis20	20	50	5	$4Q - J$	No	4	3	0
				$5Q - J$	Yes	25204809	37	403
Sanchis22	22	50	5	$4Q - J$	No	6	5	0
				$5Q - J$	Yes	57308615	39	1156
c-fat14-1	14	52	6	$5Q - J$	No	5	4	0
				$6Q - J$	Yes	328379153	56	2454
Hamming4-4	16	8	2	$2Q - J$	Yes	511	8	0
Johnson6-2-4	15	45	3	$2Q - J$	No	2	1	0
				$3Q - J$	Yes	148231	21	1
Johnson6-4-4	15	45	3	$2Q - J$	No	2	1	0
				$3Q - J$	Yes	147201	22	1
Johnson7-2-4	21	105	3	$2Q - J$	No	2	1	0
				$3Q - J$	Yes	276748639	37	4979
Keller2	16	40	2	$2Q - J$	Yes	10329	15	0

Results on matrices for Johnson's MaxClique problems

Graph	n	m	ω	A	$A \in \mathcal{C}$?	n_{simp}	max l	t, s
J8-2-4	28	210	4	$3Q - J$	No	3	2	0.00
				$4Q - J$	?	495472976	58	20000
J8-4-4	70	1855	14	$6Q - J$	No	6	5	0.01
				$7Q - J$	No	7	6	0.01
				$8Q - J$	No	8	7	0.05
				$9Q - J$	No	9	8	0.05
				$10Q - J$	No	10	9	0.03
				$11Q - J$	No	375	374	0.70
				$12Q - J$	No	12	11	0.02
				$13Q - J$	No	375	374	0.70
				$14Q - J$	?	12888179	408	20000
J16-2-4	120	5460	8	$6Q - J$	No	6	5	0.13
				$7Q - J$	No	7	6	0.14
				$8Q - J$	?	2168649	488	
J32-2-4	496	107880	16	$6Q - J$	No	6	5	18.59
				$7Q - J$	No	7	6	31.06
				$8Q - J$	No	8	7	28.91
				$9Q - J$	No	9	8	24.25
				$10Q - J$	No	10	9	47.43
				$11Q - J$	No	3421	3420	10333
				$12Q - J$	No	12	11	34.29
				$13Q - J$	?	3972	3638	20003
				$14Q - J$	?	6621	3997	20003
				$15Q - J$	No	15	14	17.61
				$16Q - J$	?	3803	3638	20005

Results on problems from the Second DIMACS Challenge

Data set	n	m	ω	ω'	max l	memory V & Q	memory v^* & q^*
Brock200_1	200	14834	21	13	1990	**1.2GB**	6.1MB
Brock200_2	200	9876	12	10	980	0.6GB	3.0MB
Brock200_3	200	12048	15	11	1330	0.8GB	4.1MB
Brock200_4	200	13089	17	13	1504	0.9GB	4.6MB
Brock400_1	400	59723	27	19	4984	**11.9GB**	30.4MB
Brock400_2	400	59786	29	18	4946	**11.8GB**	30.2MB
Brock400_3	400	59681	31	19	4919	**11.7GB**	30.0MB
Brock400_4	400	59765	33	19	4978	**11.9GB**	30.4MB
Brock800_1	800	207505	23	16	7609	**72.6GB**	92.9MB
Brock800_2	800	208166	24	16	7457	**71.1GB**	91.0MB
Brock800_3	800	207333	25	13	8193	**78.1GB**	100.0MB
Brock800_4	800	207643	26	13	7432	**70.9GB**	90.7MB
Hamming6-2	64	1824	32	32	788	49.3MB	0.8MB
Hamming6-4	64	704	4	4	112	7.0MB	0.1MB
Hamming8-2	256	31616	128	128	11747	**11.5GB**	45.9MB
Hamming8-4	256	20864	16	16	1557	**1.5GB**	6.1MB
Hamming10-2	1024	518656	512	196	4910	**76.7GB**	76.7MB
Hamming10-4	1024	434176	40	32	470	**7.3GB**	7.3MB
Johnson8-2-4	28	210	4	4	58	0.7MB	25.4KB
Johnson8-4-4	70	1855	14	14	408	30.5MB	0.4MB
Johnson16-2-4	120	5460	8	8	488	0.1GB	0.9MB
Johnson32-2-4	496	107880	16	16	3997	**14.7GB**	30.3MB
Keller4	171	9435	11	8	794	0.3GB	2.1MB
Keller5	776	225990	27	15	3646	**32.7GB**	43.2MB
Keller6	3361	4619898	$\geq$59	28	139	**23.4GB**	7.1MB
MANN_a9	45	918	16	16	323	10.0MB	0.2MB
MANN_a27	378	70551	126	121	23055	**49.1GB**	133.0MB

Reformulation of conditions in copositivity detection

- Observe that for the problem $\max\{y : \ Q - yD \in \mathcal{C}\}$, $A = Q - yD$ with copositive D, the condition $v_i^T A v_j \geq 0$ can be rewritten as

$$y \leq \frac{v_i^T Q v_j}{v_i^T D v_j} : \ v_i^T (Q - yD) v_j = v_i^T Q v_j - y v_i^T D v_j.$$

- Therefore, the matrix A is not copositive, if

$$y > \frac{v^T Q v}{v^T D v}$$

for one vertex v of one of the simplices in the partition.

- Moreover, the matrix $Q - (y - \varepsilon)D$ is copositive if

$$y - \varepsilon \leq \frac{v_i^T Q v_j}{v_i^T D v_j}$$

for all vertices v_i, v_j of all simplices in the partition $\mathcal{P}$.

Reformulation for the standard quadratic programming

- In the case of the standard quadratic programming $\max\{y : \ Q - yJ \in \mathcal{C}\}$, $A = Q - yJ$, and the condition $v_i^T A v_j \geq 0$ can be rewritten as $y \leq v_i^T Q v_j$, since $v_i^T J v_j = 1$ for $v_i, v_j \in \Delta^S$.
- Therefore, the matrix A is not copositive, if $y > v^T Q v$ for one vertex v of one of the simplices.
- The matrix $Q - (y - \varepsilon)J$ is copositive if $y - \varepsilon \leq v_i^T Q v_j$ for all vertices v_i, v_j of all simplices in the partition $\mathcal{P}$.

Reformulation for the maximum clique problem

- In the case of the maximum clique problem $\min\{t : \ tQ - J \in \mathcal{C}\}$, $A = tQ - J$, and the condition $v_i^T A v_j \geq 0$ can be rewritten as $t \geq 1/(v_i^T Q v_j)$: $v_i^T (tQ - J) v_j = t v_i^T Q v_j - v_i^T J v_j = t v_i^T Q v_j - 1$, Q and v does not have negative entries.
- Therefore, the matrix A is not copositive, if $t < 1/(v^T Q v)$ for one vertex v of one of the simplices.
- The matrix $(t + \varepsilon)Q - J$ is copositive if $t + \varepsilon \geq 1/(v_i^T Q v_j)$ for all vertices v_i, v_j of all simplices in the partition $\mathcal{P}$. For the maximum clique problem $\varepsilon = 1$ is set, as this is exactly the tolerance needed for the integer clique number.
- If the previous condition holds, $\arg\max_v 1/(v^T Q v)$ defines the solution of the underlying problem (non zero values of v mean the node is in the clique), where v is a vertex of a simplex.

Sketch of the algorithm for $\max\{y : \ Q - yD \in \mathcal{C}\}$, $\max\{y : \ Q - yJ \in \mathcal{C}\}$, $\min\{t : \ tQ - J \in \mathcal{C}\}$

Start with the standard simplex
while not stopped **do**
$y \leftarrow \min\left\{y, \frac{v_i^T Q v_i}{v_i^T D v_i}\right\}$, $\min\left\{y, v_i^T Q v_i\right\}$, $t \leftarrow \max\left\{t, \frac{1}{v_i^T Q v_i}\right\}$,
$i = 1, \ldots, n$
if $y - \varepsilon \leq \frac{v_i^T Q v_j}{v_i^T D v_j}$, $y - \varepsilon \leq v_i^T Q v_j$, $t + \varepsilon \geq \frac{1}{v_i^T Q v_j}$,
$i, j = 1, \ldots, n$ **then**
restore vertices, change vertex or stop the cycle
remember the changed vertex
else
subdivide the edge with the smallest $\frac{v_i^T Q v_j}{v_i^T D v_j}$, $v_i^T Q v_j$, $v_i^T Q v_j$
remember the changed vertex
end if
end while

Experimental results on example problems

from (Bomze & De Klerk, 2002)

	$\max\{y:\ Q - yD \in \mathcal{C}\}$, $\varepsilon = 10^{-6}$				$\max\{y:\ Q - yJ \in \mathcal{C}\}$, $\varepsilon = 10^{-6}$			
Q	y^*	n_{simp}	max l	t, s	y^*	n_{simp}	max l	t, s
Q_1	0.5	19	4	0.0	0.5	19	4	0.0
Q_2	0.3333	71679	22	0.53	0.3333	71679	22	0.35
Q_3	-16.3333	23	6	0.0	-16.3333	23	6	0.0
Q_4	0.4839	89	13	0.0	0.4839	89	13	0.0

	$\min\{t:\ tQ - J \in \mathcal{C}\}$, $\varepsilon = 1$			
Q	t^*	n_{simp}	max l	t, s
Q_1	2	19	4	0.0
Q_2	3	31301	21	0.18

$$Q_1 = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}, \quad Q_3 = \begin{pmatrix} -14 & -15 & -16 & 0 & 0 \\ -15 & -14 & -12.5 & -22.5 & -15 \\ -16 & -12.5 & -10 & -26.5 & -16 \\ 0 & -22.5 & -26.5 & 0 & 0 \\ 0 & -15 & -16 & 0 & -14 \end{pmatrix},$$

$$Q_4 = \begin{pmatrix} 0.9044 & 0.1054 & 0.5140 & 0.3322 & 0 \\ 0.1054 & 0.8715 & 0.7385 & 0.5866 & 0.9751 \\ 0.5140 & 0.7385 & 0.6936 & 0.5368 & 0.8086 \\ 0.3322 & 0.5866 & 0.5368 & 0.5633 & 0.7478 \\ 0 & 0.9751 & 0.8086 & 0.7478 & 1.2932 \end{pmatrix}$$

Experimental results on generated MaxClique problems

Graph	n	m	ω	t^*	n_{simp}	max l	time, s
Brock14	14	51	5	5	1873355	33	15.73
Brock16	16	59	5	5	3929357	36	43.71
Brock18	18	78	5	5	100442543	50	1487.45
Brock20	20	95	5	5	936595215	52	18154.00
Jagota14	14	31	6	6	94425	24	0.71
Jagota16	16	57	8	8	14659409	43	157.13
Jagota18	18	84	10	10	2366989393	67	34478.00
Morgen14	14	50	5	5	2565871	37	20.97
Morgen16	16	59	5	5	1963895	36	21.39
Morgen18	18	60	5	5	10032327	38	141.71
Morgen20	20	67	5	5	298940103	43	5925.00
Morgen22	22	68	5	5	91131959	43	2109.27
Morgen24	24	69	5	5	80165637	45	2280.00
Sanchis14	14	50	5	5	1184933	31	10.09
Sanchis16	16	50	5	5	560033	28	6.05
Sanchis18	18	50	5	5	2578523	28	35.80
Sanchis20	20	50	5	5	17294981	33	315.58
Sanchis22	22	50	5	5	40407507	34	905.79
Sanchis24	24	50	5	5	172426589	38	4885.00
c-fat14-1	14	52	6	6	29692167	39	250.27
c-fat16-1	16	69	7	7	2071185519	52	23524.00
c-fat18-1	18	72	6	6	7515802065	53	109344.00
Hamming4-4	16	8	2	2	511	8	0.00
Johnson6-2-4	15	45	3	3	88483	21	0.82
Johnson6-4-4	15	45	3	3	90013	21	0.85
Johnson7-2-4	21	105	3	3	123008083	33	2829.00
Keller2	16	40	2	2	10329	15	0.10

Results for the DIMACS benchmark problems

Graph	n	m	ω	ω'	t'	n_{simp}	max l	t, s
Brock200_1	200	14834	21	13	16	708261	1989	10^4
Brock200_2	200	9876	12	10	10	710547	980	10^4
Brock200_3	200	12048	15	11	11	699520	1330	10^4
Brock200_4	200	13089	17	13	14	708598	1504	10^4
Brock400_1	400	59723	27	19	20	18115	4984	10^4
Brock400_2	400	59786	29	18	20	17955	4946	10^4
Brock400_3	400	59681	31	19	20	18027	4920	10^4
Brock400_4	400	59765	33	19	20	18162	4978	10^4
Brock800_1	800	207505	23	16	17	19290	8376	10^5
Brock800_2	800	208166	24	16	17	19317	8482	10^5
Brock800_3	800	207333	25	13	17	19275	8385	10^5
Brock800_4	800	207643	26	13	16	19379	8389	10^5
Hamm6-2_	64	1824	32	32	32	15805366	883	10^4
Hamm6-4	64	704	4	4	4	16612788	112	10^4
Hamm8-2	256	31616	128	128	128	128307	14441	10^4
Hamm8-4	256	20864	16	16	16	128632	1628	10^4
Hamm10-2	1024	518656	512	196	512	3899	3899	10^5
Hamm10-4	1024	434176	40	32	33	3857	3857	10^5
John8-2-4	28	210	4	4	4	230691654	56	10^4
John8-4-4	70	1855	14	14	14	14850322	405	10^4
John16-2-4	120	5460	8	8	8	3202090	488	10^4
John32-2-4	496	107880	16	16	16	7818	4308	10^4
Keller4	171	9435	11	8	8	696414	794	10^4
Keller5	776	225990	27	15	20	19377	8608	10^5
Keller6	3361	4619898	$\geq$59	28	37	1493	1493	10^6
MANN_a9	45	918	16	16	16	57018692	320	10^4
MANN_a27	378	70551	126	121	121	242995	23055	10^5
MANN_a45	1035	533115	345		336	74763	74763	10^6
MANN_a81	3321	5506380	$\geq$1100		302	1550	1550	10^6

Спасибо за внимание

Thank you for your attention