



LOBACHEVSKY  
UNIVERSITY

# Comparison of Dimensionality Reduction Schemes for Parallel Global Optimization Algorithms

**Konstantin Barkalov, Vladislav Sovrasov and Ilya Lebedev**

Lobachevsky State University of Nizhni Novgorod

25 September 2018  
Moscow, Russia

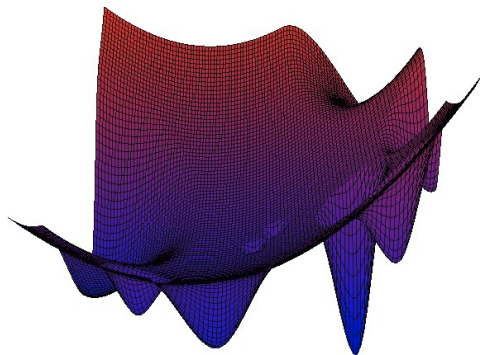
## Problem statement

$$\varphi(y^*) = \min\{\varphi(y) : y \in D\},$$
$$D = \{y \in \mathbb{R}^N : a_i \leq y_i \leq b_i, 1 \leq i \leq N\}$$

$\varphi(y)$  is multiextremal objective function,  
which satisfies the Lipschitz condition:

$$|\varphi(y_1) - \varphi(y_2)| \leq L\|y_1 - y_2\|, y_1, y_2 \in D,$$

where  $L > 0$  is the Lipschitz constant, and  
 $\|\cdot\|$  denotes  $l_2$  norm in  $\mathbb{R}^N$  space.



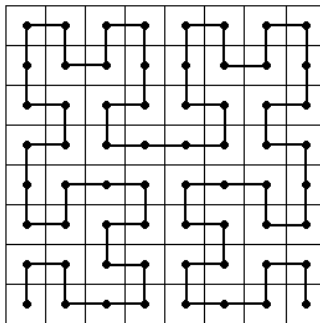
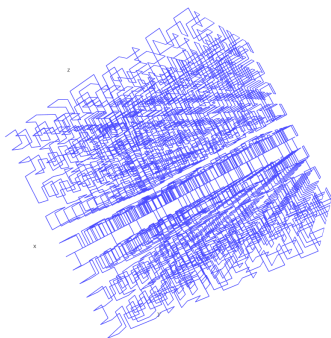
# Dimension reduction

Peano-type curve  $y(x)$  allows to reduce the dimension of the original problem:

$$\{y \in \mathbb{R}^N : -2^{-1} \leq y_i \leq 2^{-1}, 1 \leq i \leq N\} = \{y(x) : 0 \leq x \leq 1\}$$

$$\min\{f(y) : y \in D\} = \min\{f(y(x)) : x \in [0, 1]\}$$

$y(x)$  is non-smooth function which continuously maps the segment  $[0, 1]$  to the hypercube  $D$ .

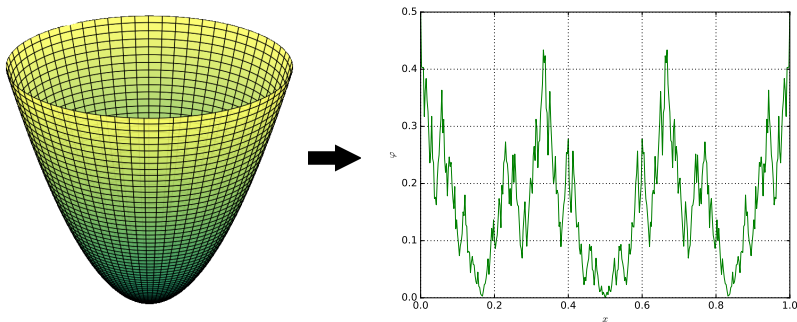


## Properties of the reduced problem

After applying the Peano-type evolvant  $\varphi(y(x))$  satisfies the uniform Hölder condition:

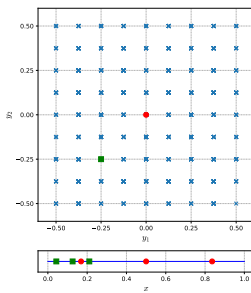
$$|\varphi(y(x_1)) - \varphi(y(x_2))| \leq H|x_1 - x_2|^{\frac{1}{N}}, x_1, x_2 \in [0, 1],$$

$\varphi(y(x))$  is non-smooth and has multiple local and **global** extremums even if  $\varphi(y)$  is unimodal. The latter problem is caused by loss of the information about  $N$ -d neighborhood after the transformation to the 1-d space.



## Non-univalent evolver

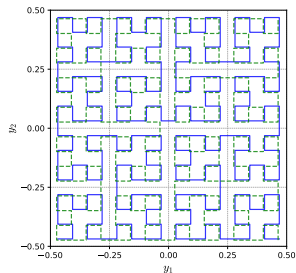
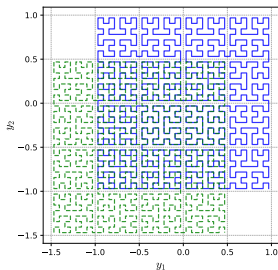
One can try to recover all preimages of  $y \in \mathbb{R}^N$  and make optimization method aware of their existence<sup>1</sup>. This allows reducing the effect of growing amount of local minimas after dimension reduction. According to the theory of Peano-type curves, each  $N$ -d point could have up to  $2^N$  preimages. For large  $N$  such preimages mining would be expensive.



<sup>1</sup>R.G. Strongin. Numerical Methods in Multiextremal Problems (in Russian), 1978

## Shifted and rotated evolvents

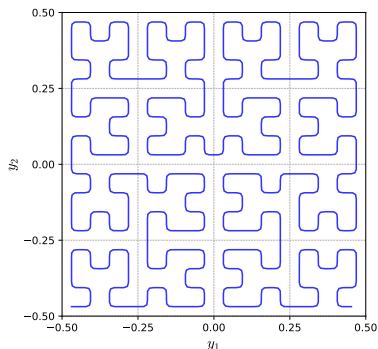
To create a fixed amount of preimages one can use a pre-defined set of different evolvents. These evolvents could be shifted or rotated versions of the original one. Set of shifted evolvents<sup>2</sup> is theoretically proven to generate at least one pair of close preimages if images are close and it perform better than the set of rotated curves.



<sup>2</sup>Strongin, R.G., Gergel, V.P., Barkalov, K.A. Parallel methods for global optimization problem solving (in Russian), 2009

## Smooth evolvent

Smooth functions are more predictable for optimizer, so smooth approximation of the Peano-like  $y(x)$  curve could improve convergence rate <sup>3</sup>.



---

<sup>3</sup>Goryachih, A. A class of smooth modification of space-filling curves for global optimization problems, NET 2016

# Basic parallel optimization method

Optimization method generates search sequence  $\{x_k\}$  and consists of the following steps:

**Step 1.** Sort the search information (one-dimensional points) in increasing order.

**Step 2.** Compute the evolvent  $y(x^{k+j})$  and the function  $\varphi(y(x^{k+j}))$ ,  $j = \overline{1, p}$ .

**Step 3.** For each interval  $(x_{i-1}, x_i)$  compute quantity  $R(i)$ , called characteristic.

**Step 4.** Choose  $p$  intervals  $(x_{t_j-1}, x_{t_j})$  with the greatest characteristics and compute objective  $f(y(x^{k+j}))$  in points chosen using the decision rule  $d$ :

$$x^{k+1+j} = d(t) \in (x_{t_j-1}, x_{t_j}), j = \overline{1, p}$$

**Step 5.** If  $x_{t_j} - x_{t_j-1} < \varepsilon$  for one of  $j = \overline{1, p}$ , stop the method.

*Detailed description: Strongin R.G., Sergeyev Ya.D.: Global optimization with non-convex constraints. Sequential and parallel algorithms (2000), Chapter 7*



## Parallel optimization method with multiple evolvents

Using the multiple mapping allows solving initial problem by parallel solving the problems

$$\min\{\varphi(y^s(x)) : x \in [0, 1]\}, 1 \leq s \leq S$$

on a set of intervals  $[0, 1]$  by the index method. Each one-dimensional problem is solved on a separate processor. The trial results at the point  $x^k$  obtained for the problem being solved by particular processor are interpreted as the results of the trials in the rest problems (in the corresponding points  $x^{k_1}, \dots, x^{k_S}$ ). In this approach, a trial at the point  $x^k \in [0, 1]$  executed in the framework of the  $s$ -th problem, consists in the following sequence of operations:

- Step 1. Determine the image  $y^k = y^s(x^k)$  for the evolvent  $y^s(x)$ .
- Step 2. Inform the rest of processors about the start of the trial execution at the point  $y^k$  (the blocking of the point  $y^k$ ).
- Step 3. Determine the preimages  $x^{k_s} \in [0, 1], 1 \leq s \leq S$ , of the point  $y^k$  and interpret the trial executed at the point  $y^k \in D$  as the execution of the trials in the  $S$  points  $x^{k_1}, \dots, x^{k_S}$ .
- Step 4. Inform the rest of processors about the trial results at the point  $y^k$ .

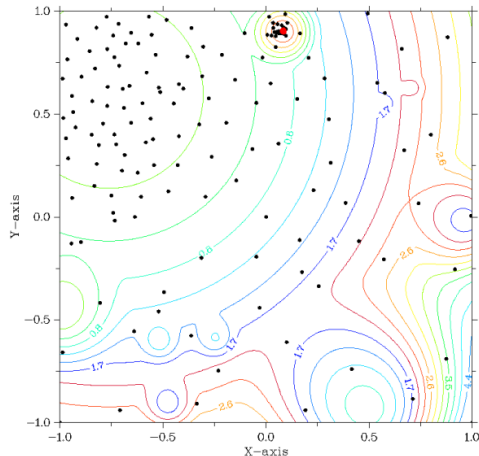
# Test problems

Generator GKLS was employed to construct the sets of test problems:

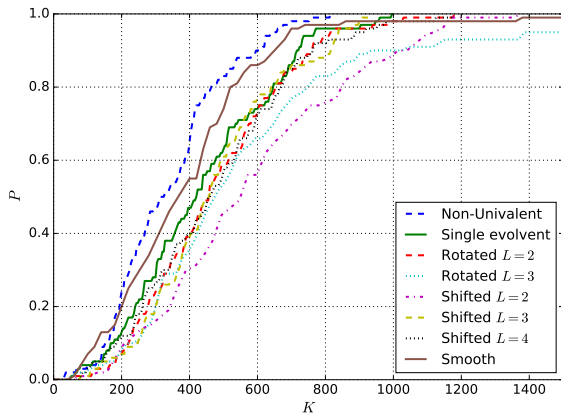
$$f(x) = \begin{cases} C_i(x), & x \in S_i, i \in 2, \dots, m \\ \|x - T\|^2 + t, & x \notin S_2, \dots, S_m \end{cases}$$

The generator allows to adjust:

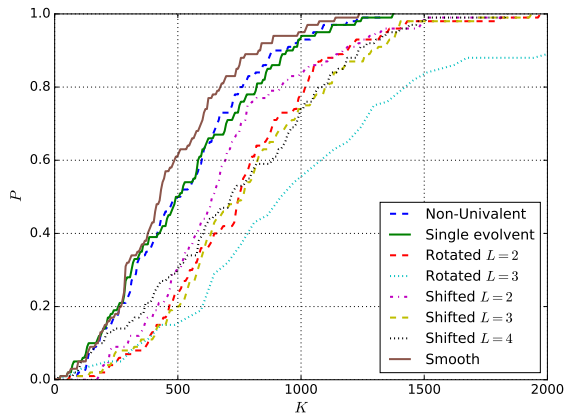
- ▶ the number of local minimas;
- ▶ the size of the global minima attraction region;
- ▶ the space dimension.



# Evolvents comparison



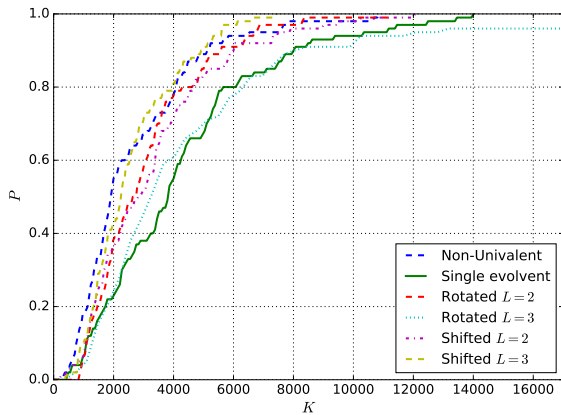
Minimal  $r$



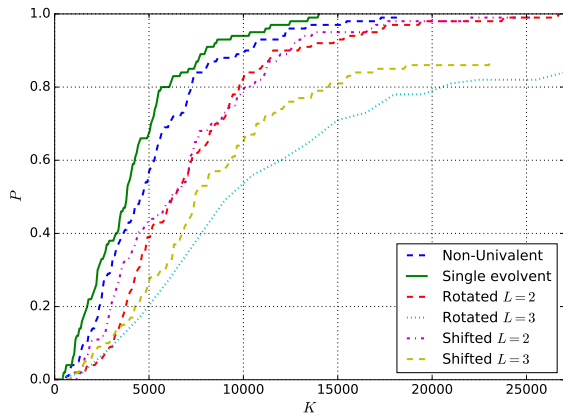
$r = 5.0$

Operating characteristics on GKLS 2d Simple class

# Evolvents comparison



Minimal  $r$



$r = 4.5$

Operating characteristics on GKLS 3d Simple class

## Choice of evolvent for the parallel algorithm

**Table:** Averaged number of computations of  $g_0$  and of  $\varphi$  when solving the problems from GKLS 3d Simple class using the shifted evolvent

$L$	$calc(g_0)$	$calc(\varphi)$	$\frac{calc(g_0)}{calc(\varphi)}$ ratio
2	96247.9	6840.14	14.07
3	153131.0	7702.82	19.88

## Results of applying the parallel algorithm

**Table:** Averaged numbers of iterations executed by the parallel algorithm for solving the test optimization problems

		$p$	$N = 4$		$N = 5$	
			<i>Simple</i>	<i>Hard</i>	<i>Simple</i>	<i>Hard</i>
I	<b>1 cluster node</b>	1	12167	25635	20979	187353
		32	328	1268	898	12208
II	<b>4 cluster nodes</b>	1	25312	11103	1472	17009
		32	64	913	47	345
III	<b>8 cluster nodes</b>	1	810	4351	868	5697
		32	34	112	35	868

## Results of applying the parallel algorithm

Table: Speedup of parallel computations executed by the parallel algorithm

		$p$	$N = 4$		$N = 5$	
			<i>Simple</i>	<i>Hard</i>	<i>Simple</i>	<i>Hard</i>
I	<b>1 cluster node</b>	1	12167(10.58s)	25635(22.26s)	20979(22.78s)	187353(205.83s)
		32	37.1(18.03)	20.2(8.55)	23.3(8.77)	15.4(9.68)
II	<b>4 cluster nodes</b>	1	0.5(0.33)	2.3(0.86)	14.3(6.61)	11.0(6.06)
		32	190.1(9.59)	28.1(1.08)	446.4(19.79)	543.0(43.60)
III	<b>8 cluster nodes</b>	1	15.0(6.05)	5.9(2.36)	24.2(17.56)	32.9(24.87)
		32	357.9(2.36)	228.9(2.64)	582.8(20.96)	793.0(33.89)

## Conclusions

- ▶ the smooth evolver and the non-univalent one demonstrate the best result in the problems of small dimensionality and can be applied successfully in solving the problems with the computationally costly objective functions.
- ▶ the shifted evolvers introduce large overhead costs on the execution of the method due to the requirement to adding an auxiliary constraint. About 95% of iterations are overhead to fight the auxiliary constraint.
- ▶ rotated evolvers perform almost the same as the shifted ones but without overhead.
- ▶ parallel optimization method shows up to 33x speedup on hard  $5d$  problems when using a set of rotated evolvers.



Contacts:

`sovrasov.vlad@gmail.com`

`https://github.com/sovrasov`