Comparison of dimensionality reduction schemes for parallel global optimization algorithms *

Konstantin Barkalov, Vladislav Sovrasov, and Ilya Lebedev

Lobachevsky State University of Nizhny Novgorod, Nizhny Novgorod, Russia
konstantin.barkalov@itmm.unn.ru
sovrasov.vlad@gmail.com
ilya.lebedev@itmm.unn.ru
http://hpc-education.unn.ru/основные-направления/глобальная-оптимизация

Аннотация This work considers a parallel algorithms for solving multi-extremal optimization problems. Algorithms are developed within the framework of the information-statistical approach and implemented in a parallel solver "Globalizer" . The optimization problem is solved by reducing the multidimensional problem to a set of joint one-dimensional problems that are solved in parallel. Five types of Peano-type space-filling curves are employed to reduce dimension. The results of computational experiments carried out on several hundred test problems are discussed.

Keywords: Global optimization · Dimension reduction · Parallel algorithms · Multidimensional multiextremal optimization · Global search algorithms · Parallel computations

1 Introduction

2 Statement of Multidimensional Global Optimization Problem

In this paper, the core class of optimization problems, which can be solved using Globalizer[12], is formulated. This class involves the multidimensional global optimization problems without constraints, which can be defined in the following way:

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

with the given boundary vectors a and b. It is supposed, that the objective function $\varphi(y)$ satisfies the Lipschitz condition

$$|\varphi(y_1) - \varphi(y_2)| \le L\|y_1 - y_2\|, y_1, y_2 \in D, \tag{2}$$

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

* The study was supported by the Russian Science Foundation, project No 16-11-10150

Usually, the minimized function $\varphi(y)$ is defined as a computational procedure, according to which the value $\varphi(y)$ can be calculated for any vector $y \in D$ (let us further call such a calculation *a trial*). It is supposed that this procedure is a time-consuming one. As a result, the overall time of solving the optimization problem (1) is determined, first of all by the number of executed trials. It should also be noted that the requirement of the Lipschitz condition (2) is highly important, since an estimate of the global minimum can be constructed on the basis of a finite number of computed values of the optimized function only in this case .

3 Methods of Dimension Reduction

3.1 Single evolvent

Within the framework of the information-statistical global optimization theory, the Peano space-filling curves (or evolvents) $y(x)$ mapping the interval $[0,1]$ onto an N-dimensional hypercube D unambiguously are used for the dimensionality reduction [1], [2], [4], [5].

As a result of the reduction, the initial multidimensional global optimization problem (1) is reduced to the following one-dimensional problem:

$$\varphi(y(x^*)) = \min\{\varphi(y(x)) : x \in [0,1]\}. \tag{3}$$

It is important to note that this dimensionality reduction scheme transforms the minimized Lipschitzian function from (1) to the corresponding one-dimensional function $\varphi(y(x))$, which satisfies the uniform Hölder condition, i. e.

$$|\varphi(y(x_1)) - \varphi(y(x_2))| \leq H|x_1 - x_2|^{\frac{1}{N}}, x_1, x_2 \in [0,1], \tag{4}$$

where the constant H is defined by the relation $H = 2L\sqrt{N+3}$, L is the Lipschitz constant from (2), and N is the dimensionality of the optimization problem (1).

The algorithms for the numerical construction of the Peano curve approximations are given in [5].

The computational scheme obtained as a result of the dimensionality reduction consists of the following:

- The optimization algorithm performs the minimization of the reduced one-dimensional function $\varphi(y(x))$ from (3),
- After determining the next trial point x, a multidimensional image y is calculated by using the mapping $y(x)$,
- The value of the initial multidimensional function $\varphi(y)$ is calculated at the point $y \in D$,
- The calculated value $z = \varphi(y)$ is used further as the value of the reduced one-dimensional function $\varphi(y(x))$ at the point x.

3.2 Shifted evolvents

The reduction of the multidimensional problems to the one-dimensional ones using the Peano curves has such important properties as the continuity and

the uniform boundedness of the function differences for limited variation of argument. However, a partial loss of information on the nearness of the points in the multidimensional space takes place since a point $x \in [0,1]$ has the left and the right neighbors only while the corresponding point $y(x) \in D \subset R^N$ has the neighbors in $2N$ directions. As a result, when using the mappings like Peano curve the images y', y'', which are close to each other in the N-dimensional space can correspond to the preimages x', x'', which can be far away from each other in the interval $[0,1]$. This property results in the excess computations since several limit points x', x'' of the trial sequence generated by the index method in the interval $[0,1]$ can correspond to a single limit point y in the N-dimensional space.

One of the possible ways to overcome this disadvantage consists in using the multiple mappings

$$Y_L(x) = \left\{y^0(x),\ y^1(x), ...,\ y^L(x)\right\} \tag{5}$$

instead of single Peano curve $y(x)$ (see [8,3,5]).

Such set of evolvents can be produced by shifting the source evolvent $y^0(x)$ by $2^{-l}, 0 \le l \le L$ on each coordinate. Each evolvent has it's own corresponding hypercude $D_l = \left\{y \in R^N : -2^{-1} \le y_i + 2^{-l} \le 3 \cdot 2^{-1},\ 1 \le i \le N\right\},\ 0 \le l \le L$.

In Fig. 1a the image of the interval $[0,1]$ obtained by the curve $y^0(x),\ x \in [0,1]$, is shown as the dashed line. Since the hypercube D from (1) is included in the common part of the family of hypercubes D_l (the boundaries of hypercube D are highlighted in Fig. 1b), having introduced an additional constraint function

$$g_0(y) = \max\left\{|y_i| - 2^{-1} :\ 1 \le i \le N\right\}, \tag{6}$$

one can present the initial hypercube D in the form

$$D = \left\{y^l(x) :\ x \in [0,1],\ g_0(y^l(x)) \le 0\right\},\ 0 \le l \le L,$$

i.e., $g_0(y) \le 0$ if $y \in D$ and $g_0(y) > 0$ otherwise. Consequently, any point $y \in D$ has its own preimage $x^l \in [0,1]$ for each mapping $y^l(x),\ 0 \le l \le L$.

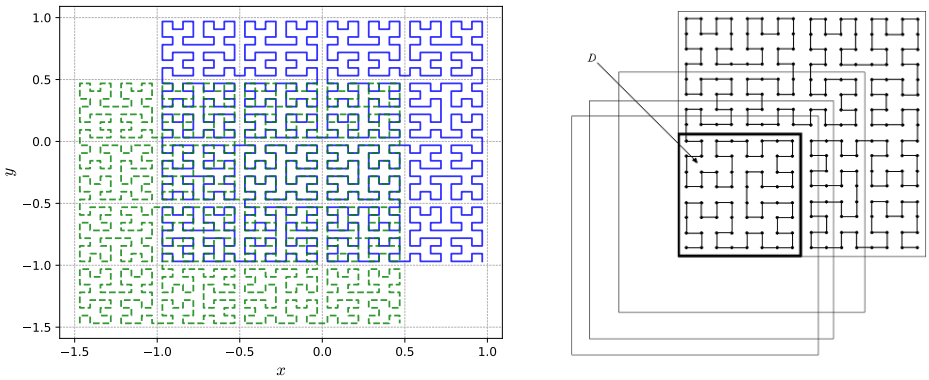
Thus, each evolvent $y^l(x),\ 0 \le l \le L$, generates its own problem of the type (1) featured by its own extended (in comparison with D) search domain D_l and the additional constraint with the left hand part from (6)

$$\min\left\{\varphi(y^l(x)) : x \in [0,1],\ g_j(y^l(x)) \le 0,\ 0 \le j \le m\right\},\ 0 \le l \le L. \tag{7}$$

3.3 Rotated evolvents

The application of the scheme for building the multiple evolvents (hereinafter called the shifted evolvents or S-evolvents) described in Subsection 3.2 allows to preserve the information on the nearness of the points in the multidimensional space and, therefore, to provide more precise (as compared to a single evolvent) estimate of Lipschitz constant in the search process. However, this approach has serious restrictions, which narrow the applicability of the parallel algorithms, designed on the base of the S-evolvents (see te end of the section 5.1).

To overcome the S-evolvent and to preserve the information on the nearness of the points in the N-dimensional space, a novel scheme of building of the multiple mappings is proposed. The building of a set of Peano curves not by the



(a) Two mappings (b) Hypercubes D_l

Рис. 1: Multiple mappings

shift along the main diagonal of the hypercube but by rotation of the evolvents around the coordinate origin is a distinctive feature of the proposed scheme [9]. In the initial non-rotated mapping for close points y', y'' in the multidimensional space their preimages x', x'' in the interval $[0, 1]$ can be far away from each other. In the rotated scheme there exists a mapping $y^i(x)$ according to which preimages x', x'' will be located nearer. The evolvents generated according to this scheme are called the rotated evolvents or R-evolvents.

In Fig. 2 two evolvents being the approximations to Peano curves for the case $N = 2$ are presented as an illustration. Taking into account the initial mapping, one can conclude that current implementation of the method allows to build up to $N(N-1)+1$ evolvents for mapping the N-dimensional domain onto the corresponding one-dimensional intervals. Moreover, the additional constraint $g_0(y) \leq 0$ with $g_0(y)$ from (6), which arises in shifted evolvents, is absent. This method for building a set of mappings can be "scaled" easily to obtain more evolvents (up to 2^N) if necessary.

3.4 Non injective evolvent

Как уже было сказано в секции 3.2, потеря информации о близости точек в многомерном пространстве может быть частично скомпенсирована использованием множественных отображений $Y_L(x) = \{y^1(x), ..., y^L(x)\}$. Однако, сама по себе кривая типа Пеано сохраняет в себе часть этой информации: она не является инъективым отображением, поэтому имея один образ $y(x) \in \mathbb{R}^N$, можно получить несколько несколько отличных x прообразов $t_j \in [0, 1], t_j \neq x$, которые затем могут быть добавлены в поисковую информацию индексного метода.

Кривая типа Пеано, используемая в (3) для редукции размерности, определяется через предельный переход, поэтому не может быть вычислена непосредственно. При численной оптимизации используется некоторое её прибли-

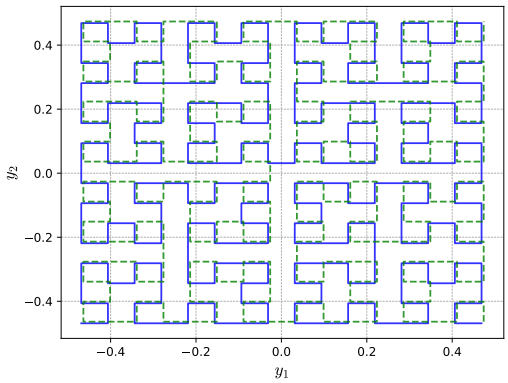


Рис. 2: Two rotated evolvents on the same plane

жение, являющееся инъективной кусочно-линейной кривой. В [2] было предложено неинъективное отображение равномерной сетки на отрезка $[0,1]$ на равномерную сетку в гиперкубе D. Каждый многомерный узел может иметь до 2^N одномерных прообразов. На рис. 3b изображена кривая с самопересечениями, полученная при соединении узлов грубой многомерной сетки в порядке следования их прообразов из отрезка $[0,1]$, а также отмечена точка, имеющая 3 прообраза.

Недостатком неинъективной развёртки является потенциально большое количетво прообразов (до 2^N) и невозможность использования параллельной схемы для множественных отображений из секции 4.2.

3.5 Smooth evolvent

4 Parallel Computations for Solving Global Optimization Problems.

4.1 Core Multidimensional Algorithm of Global Search

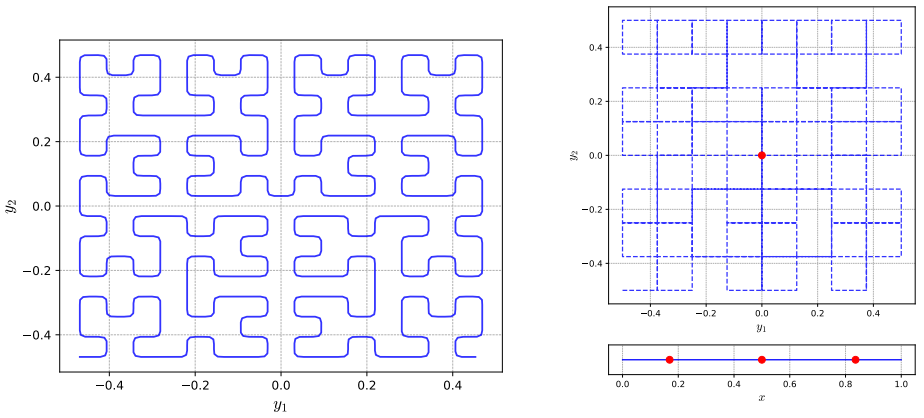
The optimization methods applied in Globalizer to solve the reduced problem (3) are based on the MAGS method, which can be presented as follows — see [2], [5].

The initial iteration of the algorithm is performed at an arbitrary point $x^1 \in (0,1)$. Then, let us suppose that k, $k \geq 1$, optimization iterations have been completed already. The selection of the trial point x^{k+1} for the next iteration is performed according to the following rules.

Rule 1. Renumber the points of the preceding trials by the lower indices in order of increasing value of coordinates $0 = x_0 < x_1 < ... < x_{k+1} = 1$.

Rule 2. Compute the characteristics $R(i)$ for each interval $(x_{i-1}, x_i), 1 \leq i \leq k+1$.

Rule 4. Determine the interval with the maximum characteristic $R(t) = \max_{1 \leq i \leq k+1} R(i)$.



(a) Smooth evolvent

(b) Numerical non injective evolvent

Рис. 3: Different evolvents

Rule 5. Execute a new trial at the point x^{k+1} located within the interval with the maximum characteristic from the previous step $x^{k+1} = d(x_t)$.

The stopping condition, which terminated the trials, is defined by the inequality $\rho_t < \varepsilon$ for the interval with the maximum characteristic from Step 4 and $\varepsilon > 0$ is the predefined accuracy of the optimization problem solution. If the stopping condition is not satisfied, the index k is incremented by 1, and the new global optimization iteration is executed.

The convergence conditions and exact formulas for descision rules $R(i)$ and $d(x)$ of the described algorithm are given, for example, in [5].

4.2 Parallel algorithm exploiting a set of evolvents

Using the multiple mapping allows solving initial problem (1) by parallel solving the problems

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

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 $x^(k_1), \ldots, x^(k_S)$). 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.

1. Determine the image $y^k = y^s(x^k)$ for the evolvent $y^s(x)$.

2. Inform the rest of processors about the start of the trial execution at the pointy^k (the blocking of the point y^k).

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 $x^{k_1}, \ldots, x^{k_s}$

4. Inform the rest of processors about the trial results at the point y^k.

The decision rules for the proposed parallel algorithm, in general, are the same as the rules of the sequential algorithm (except the method of the trial execution). Each processor has its own copy of the software realizing the computations of the problem functions and the decision rule of the index algorithm. For the organization of the interactions among the processors, the queues are created on each processor, where the processors store the information on the executed iterations in the form of the tuples: the processor number s, the trial point x^{k_s}.

The proposed parallelization scheme was implemented with the use of MPI technology. Main features of implementation consist in the following. A separate MPI-process is created for each of S one-dimensional problems being solved, usually, one process per one processor employed. Each process can use p threads, usually one thread per an accessible core.

At every iteration of the method, the process with the index $s, 0 \leqslant s < S$ performs p trials in parallel at the points $x^{(s+iS)}, 0 \leqslant i < p$. At that, each process stores all S_p points, and an attribute indicating whether this point is blocked by another process or not is stored for each point. Let us remind that the point is blocked if the process starts the execution of a trial at this point.

At every iteration of the algorithm, operating within the s-th process, determines the coordinates of p «its own» trial points. Then, the interchange of the coordinates of images of the trial points $y^{(s+iS)}, 0 \leqslant i < p, 0 \leqslant s < S$ is performed (from each process to each one). After that, the preimages $x^{(q+iS)}, 0 \leqslant q < S, q \neq s$ of the points received by the s-th process from the neighbor ones are determined with the use of the evolvent $y^s(x)$. The points blocked within the s-th process will correspond to the preimages obtained. Then, each process performs the trials at the non-blocked points, the computations are performed in parallel using OpenMP. The results of the executed trials (the index of the point, the computed values of the problem functions, and the attribute of unblocking of this point) are transferred to all rest processes. All the points are added to the search information database, and the transition to the next iteration is performed.

5 Results of Numerical Experiments

The computational experiments have been carried out on the Lobachevsky supercomputer at State University of Nizhny Novgorod. A computational node included 2 Intel Sandy Bridge E5-2660 2.2 GHz processors, 64 GB RAM. The CPUs had 8 cores (i. e. total 16 cores were available per a node). Все рассматриваемые алгортимы и развёртки были реализованы на языке C++ в рамках программной системы Globalizer[12]. Для реализации параллелизма на одном узле использована технология OpenMP, а для параллелизма на несколько узлов — стандарт MPI.

Сравнение алгоритмов глобальной оптимизации проведено путём оценки качества решения алгоритмами выборки задач из некоторого тестового класса. В данной статье рассматривается тестовый класс, порождаемый генератором GKLS [10]. Данные генератор позволяет конструировать слож-

ные многоэкстремальные задачи различной размерности. В работе рассматриваются выборки по 100 задач классов размерности 2, 3, 4, 5. Каждый класс имеет две степени сложности — *Simple* и *Hard*. Параметры генератора для рассматриваемых классов приведены в [10].

In order to evaluate the efficiency of an algorithm on a given set of 100 problems, we will use the operating characteristics [11], which are defined by a set of points on the (K, P) plane where K is the average number of search trials conducted before satisfying the termination condition when minimizing a function from a given class, and P is the proportion of problems solved successfully. If at a given K, the operating characteristic of a method goes higher than one from another method, it means that at fixed search costs, the former method has a greater probability of finding the solution. If some value of P is fixed, and the characteristic of a method goes to the left from that of another method, the former method requires fewer resources to achieve the same reliability.

5.1 Сравнение последовательных разверток

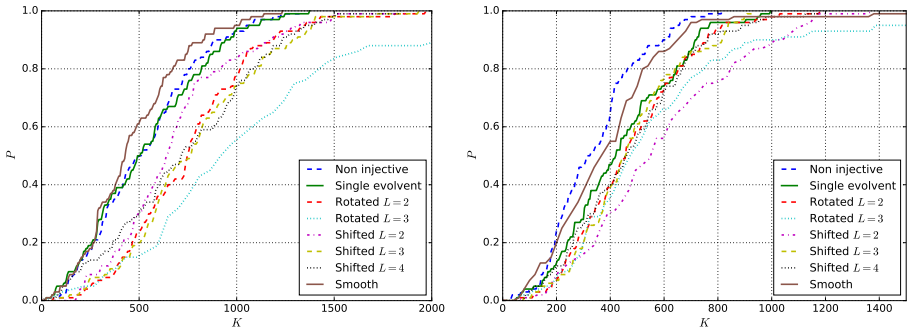
С целью понять, обладает ли какой-либо из перечисленных ранее типов развёрток существенным преимуществом над другими, были построены операционные характеристики индексного метода с различными типами развёрток на классах GKLS 2d Simple и GKLS 3d Simple. The global minimum was considered to be found if the algorithm generates a trial point y^k in the δ-vicinity of the global minimizer, i.e. $\left\|y^k - y^*\right\|_\infty \leq \delta$. The size of the vicinity was selected as $\delta = 0.01 \left\|b - a\right\|_\infty$. In case of GKLS $\delta = 0.01$.

Во всех экспериментах параметр плотности построения развёрток $m = 12$. Минимальное значение пераметра надёжности r было найдено для каждого типа развёртки перебором по равномерной сетке с шагом 0.1.

На классе GKLS 2d Simple при минимальном r неинъективная и гладкая развёртка обеспечивают более быструю сходимость (рис. 4b). То же самое, наблюдается и при $r = 5.0$ (рис. 4a). В последнем случае сдвиговая и вращаемая развёртки начинают отставать от остальных, т.к. значение $r = 5.0$ является завышенным для них.

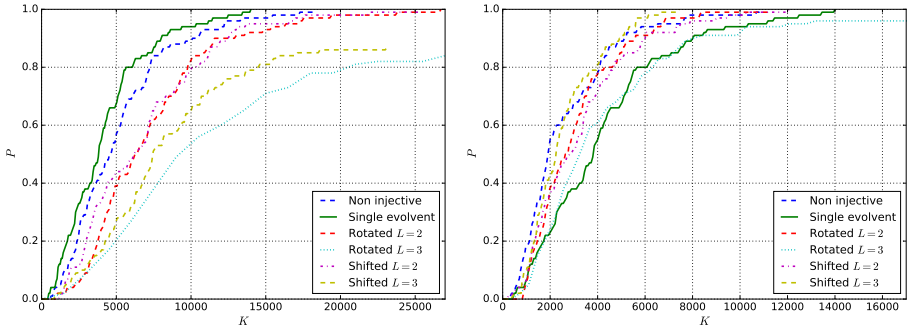
На классе GKLS 2d Simple при минимальном r неинъективная и множественные развёртки имеют значительное преимущество над единственной развёрткой (рис. 5b). Значение $r = 4.5$ является завышенным для вращаемой и сдвиговой развёрток (рис. 5a).

Накладные расходы при использовании сдвиговой развёртки. Во всех представленных выше экспериментах при построении операционной характеристики учитывалось количество вычислений целевой функции из класса GKLS, однако в случае сдвиговой развёртки индексный метод решает задачу с ограничением g_0 из (6). В точках, где g_0 нарушено, значение целевой функции не вычисляется. Эти точки, тем не менее, хранятся в поисковой информации, создавая дополнительные расходы вычислительных ресурсов.



(a) $r = 5.0$ (b) Minimal r

Рис. 4: Operating characteristics on GKLS 2d Simple class



(a) $r = 4.5$ (b) Minimal r

Рис. 5: Operating characteristics on GKLS 3d Simple class

В таблице 1 приведено среднее количество обращение к g_0 и целевой функции. При $L = 3$ ограниение g_0 вычисляется почти в 20 раз чаще, чем целевая функция φ, т. е. 95% всей поисковой информации приходится на вспомогательные точки. Такие накладные расходы приемлемы при решении задач малой размерности с трудоёмкими целевыми функциями, но при росте размерности и общего количества испытаний выгоднее использовать другие типы развёрток.

5.2 Параллельные вращаемые развертки

Чтобы оценить эффективность параллельного алгортима из секции 4.2 были проведены вычислительные эксперименты на классах GKLS 4d (Hard, Simple) и GKLS 5d (Hard, Simple). Значения r во всех экспериментах равно 5.0, размер δ-окрестности известного решения увеличен до 0.3. При решении серий задач использовалось до 8 узлов кластера и до 32 вычислительных потоков на каждом узле.

Таблица 1: Среднее количество вычислений g_0 и φ при решении задач класса GKLS 3d Simple с помощью сдвиговой развёртки

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

В таблице 2 приведено среднее количество итераций при решении 100 задач из каждого из рассматриваемых классов. При увеличении числа узлов и числа потоков на каждом узле количество итераций заметно сокращается (за исключением класса GKLS 4d Simple при переходе с 1 узла на 4 узла в однопоточном режиме).

Таблица 2: Averaged numbers of iterations executed by the parallel algorithm for solving the test optimization problems

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

Если считать, что затраты на параллелизм пренебрежимо малы по сравнению с затратами на вычисление целевых функций в задачах оптимизации, то ускорение по времени от использования параллельного метода будет равно ускорению по итерациям. Однако, в действительности это предположение не всегда справедливо. Во всех численных экспериментах время вычисления целевой функции занимает примерно 10^{-3}с. В таблице 3 приведено ускорение по итерациям и в куглых скобках ускорение по времени. В первой строке таблицы, соответствующей последовательному режиму, в скобках приведено среднее время решения одной задачи. Из таблицы видно, что для класов GKLS 4d выгоднее использовать один узел в многопоточном режиме, тогда как для решения более сложных пятимерных задач лучше использовать несколько узлов, каждый из которых работает в параллельном режиме.

Таблица 3: Speedup of parallel computations executed by the parallel algorithm

		p	$N = 4$		$N = 5$	
			Simple	*Hard*	*Simple*	*Hard*
I	**1 cluster node**	*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	**4 cluster nodes**	*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	**8 cluster nodes**	*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)

6 Conclusions

В работе рассмотрены 5 различных отображений типа кривой Пеано, применяемых для редукции размерности в задачах глобальной оптимизации. Из предварительного сравнения, проведенного в секции 5.1, можно сделать слудующие выводы:

- гладкая и неинъективная развёртки показывают наилучший результат в задачах малой размерности и могут быть успешно применены при решении задач с трудоемкими целевыми функциями. Особенности этих развёрток не позволяют создать на их основе масштабируемый на несколько узлов кластера алгоритм оптимизации.
- сдвиговые развёртки привносят большие накладные расходы на работу метода из-за того, что требуется добавления в задачу (1) вспомогательного функционального ограничения. Эксперименты показали, что до 95% поисковой информации приходится на точки, в которых вичисляется только вспомогательное ограничение. Сдвиговые развёртки могут быть использованы в качестве основы для параллельного алгоритма из секции 4.2, однако расходы на обработку дополнительных точек, скорее всего, приведут к малому коэффициенту ускорения от параллельности. Однако, при достаточно трудоёмкой целевой функции их использование может иметь смысл.
- вращаемые развёртки обеспечили приемлемую скорость сходимости на задачах малой размерности в последовательном случае. Их использование не приводит к введению дополнительных ограничений, что позволяет построить эффективный параллельный алгоритм на их основе.

В секции 5.2 приведены результаты численных экспериментов, которые показывают, что алгоритм из раздела 4.2 на основе вращаемых развёрток позволяет получить ускорение до 43 раз при решении серии задач с использованием нескольких узлов вычислитеьного кластера. Стоит заметить, что в расматриваемых задачах целевые функции не являются трудоёмкими (среднее время вычисления 10^{-3}с) и в случае более сложных задач показатели ускорения по времени будут приближаться к ускорению по итерациям.

Список литературы

1. Y. D. Sergeyev, R. G. Strongin and D. Lera, *Introduction to Global Optimization Exploiting Space-filling Curves*, Springer, 2013.
2. R. G. Strongin, *Numerical Methods in Multi-Extremal Problems (Information-Statistical Algorithms)*, Moscow: Nauka, In Russian, 1978.
3. R. G. Strongin, Algorithms for multi-extremal mathematical programming problems employing a set of joint space-filling curves, *J. Glob. Optim.*, **2** (1992), 357–378.
4. R. G. Strongin, V. P. Gergel, V. A. Grishagin and K. A. Barkalov, *Parallel Computations for Global Optimization Problems*, Moscow State University (In Russian), Moscow, 2013.
5. 5 (MR1797058) [10.1007/978-1-4615-4677-1] R. G. Strongin and Y. D. Sergeyev, *Global Optimization with Non-convex Constraints. Sequential and Parallel Algorithms*, Kluwer Academic Publishers, Dordrecht (2000, 2nd ed. 2013, 3rd ed. 2014).
6. A. Törn and A. Žilinskas, *Global Optimization*, Springer, 1989.
7. A. A. Zhigljavsky, *Theory of Global Random Search*, Kluwer Academic Publishers, Dordrecht, 1991.
8. Strongin, R.G.: Parallel multi-extremal optimization using a set of evolvents. Comp. Math. Math. Phys. **31(8)**, 37–46 (1991)
9. Strongin, R.G., Gergel, V.P., Barkalov, K.A.: Parallel methods for global optimization problem solving. Journal of instrument engineering. **52**, 25–33 (2009) (In Russian)
10. Gaviano, M., Kvasov, D.E, Lera, D., and Sergeyev, Ya.D.: Software for generation of classes of test functions with known local and global minima for global optimization. ACM Transactions on Mathematical Software 29(4), 469–480 (2003)
11. Grishagin, V.A.: Operating Characteristics of Some Global Search Algorithms. Problems of Statistical Optimization 7, 198–206 (1978) (In Russian)
12. Gergel V.P., Barkalov K.A., and Sysoyev A.V: Globalizer: A novel supercomputer software system for solving time-consuming global optimization problems. Numerical Algebra, Control & Optimization 8(1), 47–62, 2018