Investigation of selection strategies in branch and bound algorithm with simplicial partitions and combination of Lipschitz bounds

Paulavicius, R, Zilinskas, J & Grothey, A 2010, 'Investigation of selection strategies in branch and bound algorithm with simplicial partitions and combination of Lipschitz bounds' Optimization letters, vol. 4, no. 2, pp. 173-183.

Investigation of selection strategies in parallel branch and bound algorithm with simplicial partitions

Remigijus Paulavičius
Institute of Mathematics and Informatics,
Akademijos g. 4, LT-08663
Vilnius, Lithuania
Email: r.paulavicius@vpu.lt

Julius Žilinskas
Institute of Mathematics and Informatics,
Akademijos g. 4, LT-08663
Vilnius, Lithuania
Email: julius.zilinskas@mii.lt

Andreas Grothey
School of Mathematics,
University of Edinburgh,
Edinburgh EH9 3JZ, UK
Email: A.Grothey@ed.ac.uk

***Abstract*—Efficiency of parallel branch and bound algorithms depends on the selection strategy. The influence to the performance of parallel MPI branch and bound algorithm with simplicial partitions and aggregate Lipschitz bound using different selection strategy is evaluated experimentally. The experiments have been performed solving a number of multidimensional test problems for global optimization.**

I. Introduction

Many problems in engineering, physics, economics and other fields may be formulated as optimization problems, where the maximum value of an objective function must be found. Standard global optimization problem is formulated as find such

$$f^* = \max_{x \in D} f(x),$$

where the objective function $f(x), f : \mathbb{R}^n \to \mathbb{R}$, is a nonlinear function of continuous variables, $D \subset \mathbb{R}^n$ is a feasible region, n is the number of variables. Besides the global optimum f^* one or all global optimizers $x^* : f(x^*) = f^*$ must be found or shown that such a point does not exist. In our cases, D is compact and f is Lipschitz function, therefore the existence of x^* assured by the well-known theorem of Weierstrass. In Lipschitz optimization only a point $x_{opt} \in D$ such that $f(x_{opt})$ differs from f^* by no more than a specified accuracy ε can be found.

Branch and bound is a technique for the implementation of covering global optimization methods as well as combinatorial optimization algorithms. An iteration of a classical branch and bound algorithm processes a node in the search tree representing a not yet explored subspace of the solution space. The iteration has three main components: selection of the node to process, branching of the search tree and bound calculation. The rules of selection, branching and bounding differ from algorithm to algorithm.

In the branching process the algorithm detects subspaces, by evaluating bounds which cannot contain a global optimizer and discard them from further search. Although hyper-rectangular partitions are usually used in global optimization, other types of partitions may be more suitable for some specific problems. Advantages and disadvantages of simplicial partitions are discussed in [1]. Since a simplex is a polyhedron in n-dimensional space with the minimal number of vertices, simplicial partitions are preferable when the values of an objective function at the vertices of partitions are used to compute bounds. Otherwise values at some of the vertices of hyper-rectangular partitions may be used [2]. However, for simplicial branch and bound, the feasible region should be initially covered by simplices. The most preferable initial covering is face to face vertex triangulation - partitioning of the feasible region into finitely many n-dimensional simplices, whose vertices are also the vertices of the feasible region. We use a standard way [3] of triangulation into $n!$ simplices. All simplices share the diagonal of the feasible region and are of equal hypervolume. The number of initial simplices grows very fast with the dimension of the problem if such a triangulation is used therefore it can be used only when the number of variables is small. However there are problems where feasible region is either already a simplex (for example, optimization problems over the standard simplex) or may be reduced to one or a managable number of simplices (for example, when the objective function has symmetries and problems with linear constraints [4], [1]. It is also possible to over-cover the feasible region by one simplex in some cases [5]. Simplices are subdivided into two by a hyper-plane passing through the middle point of the longest edge and the vertices which do not belong to the longest edge.

The main strategies of selection are:

- *Best first* – select a candidate with maximal upper bound. The candidate list can be implemented using heap or priority queue.
- *Depth first* – select the youngest candidate. A node with the largest level in the search tree is chosen for exploration. A FILO structure is used for the candidate list which can be implemented using a stack. In some cases it is possible to implement this strategy without

storing of candidates, as it is shown in [6].

- *Breadth first* – select the oldest candidate. All nodes at one level of the search tree are processed before any node at the next level is selected. A FIFO structure is used for the candidate list which can be implemented using a queue.
- *Improved selection* – based on heuristic [7], [8], probabilistic [9] or statistical [5], [10] criteria. The candidate list can be implemented using heap or priority queue.

In this work *statistical* selection strategy [10] has been tested. Using this strategy the candidate with the maximal criterion value (1) is chosen where f^* is the global maximum or the upper bound for it.

The influence to the speed (number of function evaluations and optimization time) and memory requirements of the sequential branch and bound algorithm proposed in [11]. The goal of this paper is to experimentally investigate the influence of selection strategies to the speed and efficiency of parallel algorithms. Although the experiments have been performed on a particular algorithm described in the section III, similar features may be expected in other parallel branch and bound algorithms.

II. Lipschitz optimization

Lipschitz optimization is one of the most deeply investigated subjects of global optimization. The advantages and disadvantages of Lipschitz global optimization methods are discussed in [12], [13]. A function $f : D \to \mathbb{R},\ D \subset \mathbb{R}^n$, is said to be Lipschitz if it satisfies the condition

$$|f(x) - f(y)| \leq L \|x - y\|, \ \forall x, y \in D, \tag{2}$$

where $L > 0$ is a constant called Lipschitz constant, D is compact and $\|\cdot\|$ denotes a norm. The Euclidean norm is used most often in Lipschitz optimization, but other norms can also be considered.

If the Lipschitz bounds over a sub-regions $I \subseteq D$ are evaluated using the function values at the vertices, the lower bound (LB) for the maximal value of f is general for all sub-regions and is equal to the largest value of the function at a vertex:

$$LB(D) = \max_{v \in V(D)} f(v),$$

where $V(D) = \cup V(I)$.

The upper bound over a sub-region $I \subseteq D$ is evaluated by exploiting Lipschitz condition. It follows from (2) that, for all $x, y \in D$

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

If $y \in D$ is fixed, then the concave function

$$F_y(x) = f(y) + L\|x - y\| \tag{3}$$

overestimates $f(x)$ over D. Let T be a finite set of distinct points in D. Then, the sharpest upper bound over D, given the function values $f(y), y \in T$, and the Lipschitz constant L, is provided by

$$\varphi(D) = \max_{x \in D} \min_{y \in T} F_y(x). \tag{4}$$

In the univariate case, the function $\min_{y \in T} F_y(x)$ is piecewise linear, and φ can be determined in a simple straightforward way [12]. Therefore, many univariate algorithms use the bound φ, where the set T is suitably updated in an iterative way. The most studied of these methods is due to Piyavskii [14]. When D is a two-dimensional rectangle in $\mathbb{R}^2$, φ can still be evaluated by geometric arguments which take into account the conical shape of upper bounding function. For $(n > 2)$, however, problem (4) constitutes a difficult optimization problem.

Convergent deterministic Lipschitz optimization methods fall into three main classes. First, multivariate Lipschitz optimization can be reduced to the univariate case. Following this idea, a nested optimization scheme [14] and filling the feasible region by Peano curve [15], [16] were proposed.

The second class contains direct extensions of Piyavskii's method [14] to the multivariate case. Various modifications using the Euclidean norm [14], [17], [18] or other norms or close approximations [19], [20], [21], [22] have been proposed. Most of these algorithms can be improved by interpreting them using branch and bound method.

The third class contains many branch and bound algorithms, but, in general, considerably weaker bounds [23], [24], [25], [26], [27], [28], [29], [30]. These algorithms fit into the general framework proposed by Horst [31], Horst and Tuy [32]. The algorithms differ in the selection rules, the ways subdivision is performed and bounds are computed. In general, weaker (than φ type bound) bounds belong to the following two simple families μ_1 and μ_2. Let

$$\delta(I) = \max\{\|x - y\| : x, y \in I\}$$

denote the diameter of $I \subset D$. For example, if $I = \{x \in \mathbb{R}^n : a \leq x \leq b\}$ is an n-rectangle, then $\delta(I) = \|b - a\|$, and if I is an n-simplex, then the diameter $\delta(I)$ is the length of its longest edge. Afterwards a simple upper bound can be derived from (3):

$$\mu_1(I) = \min_{y \in T} f(y) + L\delta(I), \tag{5}$$

where $T \subset I$ is a finite sample of points in I, where the function values of f have been evaluated. If I is a rectangle or a simplex, the set T often coincides with the vertex set $V(I)$. A more tight but computationally more expensive than (5) bound is

$$\mu_2(I) = \min_{y \in T}\{f(y) + L \max_{z \in V(I)} \|y - z\|\}. \tag{6}$$

It is known that

$$|f(x) - f(y)| \leq L_p \|x - y\|_q, \tag{7}$$

where $L_p = \sup\left\{\|\nabla f(x)\|_p : x \in D\right\}$ is the Lipschitz constant, $\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right)$ is the gradient of the function $f(x)$ and $1/p + 1/q = 1,\ 1 \leq p, q \leq \infty$.

A improved μ_2 type upper bounds based on the first, Euclidean and infinite norms over a multidimensional simplex I was proposed and investigated in [33]:

$$\mu_2^{1,2,\infty}(I) = \min_{v \in V(I)} \{f(v) + K(V(I))\}, \tag{8}$$

$$\tilde{u}(I) = -\frac{\left(f^* - \frac{1}{n+1}\sum\limits_{x_v \in V(I)} f(x_v)\right)^2 - \left(\max\limits_{x_v \in V(I)} f(x_v) - \frac{1}{n+1}\sum\limits_{x_v \in V(I)} f(x_v)\right)^2}{\min\limits_{x_v \in V(I)} \left\| x_v - \frac{1}{n+1}\sum\limits_{x_v \in V(I)} x_v \right\|_2} \quad (1)$$

$$K(V(\mathbf{I})) = \min\left\{ L_\infty \max_{x \in V(\mathbf{I})} \|x - v\|_1, L_2 \max_{x \in V(\mathbf{I})} \|x - v\|_2, L_1 \max_{x \in V(\mathbf{I})} \|x - v\|_\infty \right\}. \quad (9)$$

where $K(V(\mathbf{I}))$ (9). Piyavskii φ type upper bound with the first norm was proposed in [34]:

$$\varphi^1(I) = \max_{x \in I}\left(\min_{v \in V(I)} \{ f(v) + L_\infty \|x - v\|_1 \} \right). \quad (10)$$

However the first norm does not always give the best bounds [34]. In some cases $\mu_2^{1,2,\infty}$ bound may give better results than $\varphi^1(I)$. Therefore aggregate Lipschitz bound $\varphi^1\mu_2^{2,\infty}$ was proposed in [35] and is used in this work:

$$\varphi^1\mu_2^{2,\infty}(I) = \min\left\{ \varphi^1(I), \mu_2^{2,\infty}(I) \right\}. \quad (11)$$

III. Parallel branch and bound with simplicial partitions and aggregate Lipschitz bound

A sequential branch and bound algorithm with simplicial partition and aggregate Lipschitz bound $\varphi^1\mu_2^{2,\infty}$ (11) was proposed in [35]. The parallel MPI version with static load balancing were implemented [36] using a parallel branch and bound template [37]. When the template is used, only algorithm specific rules should be described by the user and the standard parts are implemented in the template. Static load balancing is used: tasks are initially distributed evenly (if possible) among p processors. If the initial number of simplices ($n!$) is less than the number of processors, the simplices are subdivided until the number of processors is reached. Then the initial simplices are distributed. After initialization, the processors work independently and do not exchange any tasks generated later. Each parallel processor runs the same algorithm, which is shown in Algorithm 1. The algorithm is very similar to the sequential algorithm [35]. The differences are:

- Each processor covers one part of the feasible region. This is shown symbolically $\mathbb{I} = \cup\mathbb{I}^r, |\mathbb{I}^r| \approx |\mathbb{I}|/p$ using division by the number of processors.
- The best currently found value of the objective function $LB(\mathbb{I}^r)$ is local – processors do not interchange it. Comparison by efficiency and number of function evaluations criteria using MPI with interchange of the best currently found function values and without it proposed [36].
- After completion the results $\mathbf{S}^r$ of the optimization are collected.

Algorithm 1 Parallel simplicial branch and bound with aggregate Lipschitz bound

1: Cover feasible region D by $\mathbb{I} \leftarrow \{I_j | D = \cup I_j, j = 1, \ldots, n!\}$ using face-to-face vertex triangulation.
2: $\mathbb{I}$ evenly (if possible) divided among the p processors $\mathbb{I} = \cup\mathbb{I}^r, |\mathbb{I}^r| \approx |\mathbb{I}|/p$
3: $\mathbf{S}^r \leftarrow \emptyset$, $LB(\mathbb{I}^r) \leftarrow -\infty$
4: **while** ($\mathbb{I}^r \neq \emptyset$) **do**
5: Choose $I_j^r \in \mathbb{I}^r$ using selection strategy, $\mathbb{I}^r \leftarrow \mathbb{I}^r \backslash \{\mathbf{I}_j^r\}$
6: $LB(\mathbb{I}^r) \leftarrow \max\{LB(\mathbb{I}^r), \max\limits_{v \in V(I_j^r)} f(v)\}$
7: $\mathbf{S}^r \leftarrow arg\max\{f(\mathbf{S}^r), \max\limits_{v \in V(I_j^r)} f(v)\}$
8: $UB(I_j^r) = \varphi^1\mu_2^{2,\infty}(I_j^r)$
9: **if** ($UB(I_j^r) - LB(\mathbb{I}^r) > \varepsilon$) **then**
10: Branch I_j^r into 2 simplices: $I_{j_1}^r$, $I_{j_2}^r$
11: $\mathbb{I}^r \leftarrow \mathbb{I}^r \cup \{I_{j_1}^r, I_{j_1}^r\}$
12: **end if**
13: **end while**
14: Collect $\mathbf{S}^r$

IV. Experimental investigation of selection strategies in parallel branch and bound algorithm

In this section results of computational experiments are presented and discussed. Various difficult (with a solution time on a single processor of more than 1 s.; $3 \leq n \leq 6$) test problems [36] for global optimization from [12], [38], [39] have been used in our experiments. Computational experiments were performed on the parallel machine Ness at Edinburgh Parallel Computing Center (http://www.epcc.ed.ac.uk/facilities/ness/). Ness has a shared-memory architecture which allows users the option to run large threaded jobs (e.g. OpenMP) as well as message-passing (MPI) jobs. The system has two back-end X4600 symmetric multiprocessor (SMP) nodes, both containing 16 processor-cores (2.6 GHz AMD Opteron (AMD64e)) with 2GB of memory per core. Up to 16 processor-cores have been used in the experiments.

Parallel branch and bound algorithm using different selection strategies has been evaluated using the speed criteria: the number of function evaluations $(f.eval.)$, optimization time $(t(s))$ and standard criteria of parallelization: speedup $s_p = t_1/t_p$ and efficiency $e_p = s_p/p$, where t_p is time used

by the algorithm implemented on p processors.

The average numbers of function evaluations ($\overline{f.eval}$) and average execution time ($\overline{t(s)}$) for different dimensionality test problems are shown in Table I. The average number of function evaluations required for the whole optimization are similar for all selection strategies, although when number of processors increase the biggest increase of function evaluation is achieved using *depth first* selection strategy.

For all dimensionality's test problems using one processor the smallest average execution time is achieved when *depth first* and *breath first* selection strategies are used, despite the fact that sometimes the number of function evaluations is higher. A possible reason is that the time of insertion and deletion of elements to/from such a type of structure does not depend on the number of elements in the list. *Best first* and *statistical* selection strategies require prioritized list of candidates, and even with heap structure insertion is time consuming when the number of elements in the list is large. When the number of processors increase the execution time using *depth first* selection strategy is almost always bigger than when other selection strategies are used, especially when $p > 4$.

The diagrams of criteria of parallelization: speedup s_p and efficiency e_p for various numbers of processors and various dimensionality (n) of test problems using different selection strategies are shown in Figs. 1-4. The averages $\overline{s_p}$ and $\overline{e_p}$ are shown in Table II. The diagrams show that the efficiency of parallelization with different selection strategies is similar. The average efficiency of parallelization is very similar when *best first* and *statistical* selection strategies are used. The best efficiency of parallelization (especially when $p \geq 4$) is experienced when *breadth first* and the worst when *depth first* selection strategy is used. Using all selection strategies the efficiency of parallelization decreases less when the number of processors is increased for difficult test problems $(n \geq 5)$ compared with simpler test problems.

V. Conclusion

In this paper the speed and efficiency of parallelization of parallel branch and bound algorithm has been tested and compared for different selection strategies (*best first*, *statistical*, *depth first* and *breadth first*).

The number of function evaluations required for the whole optimization are similar for all selection strategies, although the *depth first* selection strategy requires the largest number of function evaluations.

The smallest optimization time with $p \leq 4$ is achieved when *depth first* and *breath first* selection strategies are used. When $p > 4$ the optimization time with *depth first* selection strategy is almost always bigger than with other selection strategies. However the influence is less significant for expensive test problems $n \geq 5$ which take longer to evaluate.

The efficiency of parallelization is similar when *best first*, *statistical* and *breadth first* selection strategies are used. The efficiency of parallelization is worst when *depth first* selection strategy is used. The efficiency of parallelization is better for difficult test problems.

Acknowledgment

The work was funded by the Research Council of Lithuania (project number MIP-108/2010). The work was partially supported by the COST Action Open European Network for High Performance Computing on Complex Environments IC0805.

References

[1] J. Žilinskas, "Branch and bound with simplicial partitions for global optimization," *Mathematical Modelling and Analysis*, vol. 13, no. 1, pp. 145–159, 2008.

[2] Y. D. Sergeyev and D. E. Kvasov, "Global search based on efficient diagonal partitions and a set of Lipschitz constants," *SIAM Journal on Optimization*, vol. 16, no. 3, pp. 910–937, 2006.

[3] M. J. Todt, ser. Lecture Notes in Economics and Mathematical Systems, 1976, vol. 24.

[4] J. Žilinskas, "Reducing of search space of multidimensional scaling problems with data exposing symmetries," *Information Technology and Control*, vol. 36, no. 4, pp. 377–382, 2007.

[5] A. Žilinskas and J. Žilinskas, "Global optimization based on a statistical model and simplicial partitioning," *Computers & Mathematics with Applications*, vol. 44, no. 7, pp. 957–967, 2002.

[6] ——, "Branch and bound algorithm for multidimensional scaling with city-block metric," *Journal of Global Optimization*, vol. 43, no. 2–3, pp. 357–372, 2009.

[7] V. Kreinovich and T. Csendes, "Theoretical justification of a heuristic subbox selection criterion for interval global optimization," *Central European Journal of Operations Research*, vol. 9, no. 3, pp. 255–265, 2001.

[8] T. Csendes, "Generalized subinterval selection criteria for interval global optimization," *Numerical Algorithms*, vol. 37, no. 1–4, pp. 93–100, 2004.

[9] M. Dür and V. Stix, "Probabilistic subproblem selection in branch-and-bound algorithms," *Journal of Computational and Applied Mathematics*, vol. 182, no. 1, pp. 67–80, 2005.

[10] A. Žilinskas and J. Žilinskas, "P-algorithm based on a simplicial statistical model of multimodal functions," *TOP*, no. submitted, 2009.

[11] R. Paulavičius, J. Žilinskas, and A. Grothey, "Investigation of selection strategies in branch and bound algorithm with simplicial partitions and combination of Lipschitz bounds," *Optimization Letters*, vol. 4, no. 2, pp. 173–183, 2010.

[12] P. Hansen and B. Jaumard, "Lipshitz optimization," in *Handbook of Global Optimization*, R. Horst and P. M. Pardalos, Eds. Kluwer Academic Publishers, 1995, pp. 407–493.

[13] R. Horst, P. M. Pardalos, and N. V. Thoai, *Introduction to Global Optimization*. Kluwer Academic Publishers, 1995.

[14] S. A. Piyavskii, "An algorithm for finding the absolute extremum of a function," *Zh. Vychisl. Mat. mat. Fiz*, vol. 12, no. 4, pp. 888–896, 1972.

[15] A. Butz, "Space filling curves and mathematical programming," *Information and Control*, vol. 12, pp. 319–330, 1968.

[16] R. Strongin, "Algorithms for multi-extremal mathematical programming problems employing the set of joint space-filling curves," *Journal of Global Optimization*, vol. 2, pp. 357–378, 1992.

[17] R. H. Mladineo, "An algorithm for finding the global maximum of a multimodal, multivariate function," *Mathematical Programming*, vol. 34, no. 2, pp. 188–200, 1986.

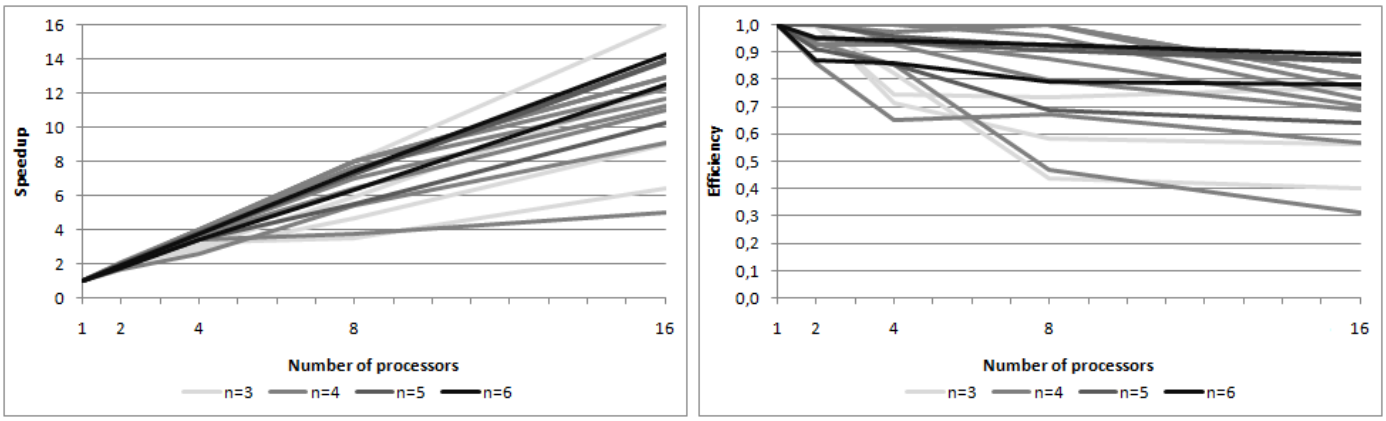
[18] B. Jaumard and T. H. ant H. Ribault, "An on-line cone intersection algorithm for global optimization of multivariate lipschitz functions," *Les Cahiers du GERAD*, vol. 95, no. 7, 1995.

[19] D. Q. Mayne and E. Polak, "Outer approximation algorithm for nondifferentiable optimization problems," *Journal of Optimization Theory and Applications*, vol. 42, no. 1, pp. 19–30, 1984.

[20] G. R. Wood, "Multidimensional bisection applied to global optimisation," *Computers & Mathematics with Applications*, vol. 21, no. 6-7, pp. 161–172, 1991.

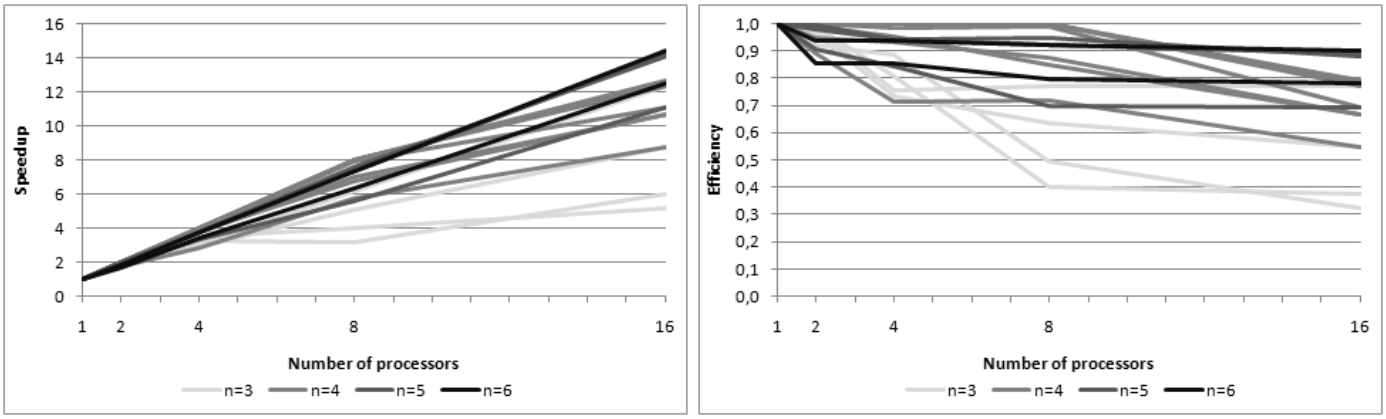
[21] ——, "The bisection method in higher dimensions," *Mathematical Programming*, vol. 55, pp. 319–337, 1992.

[22] B. Zhang, G. Wood, and W. Baritompa, "Multidimensional bisection: The performance and the context," *Journal of Global Optimization*, vol. 3, no. 3, pp. 337–358, 1993.



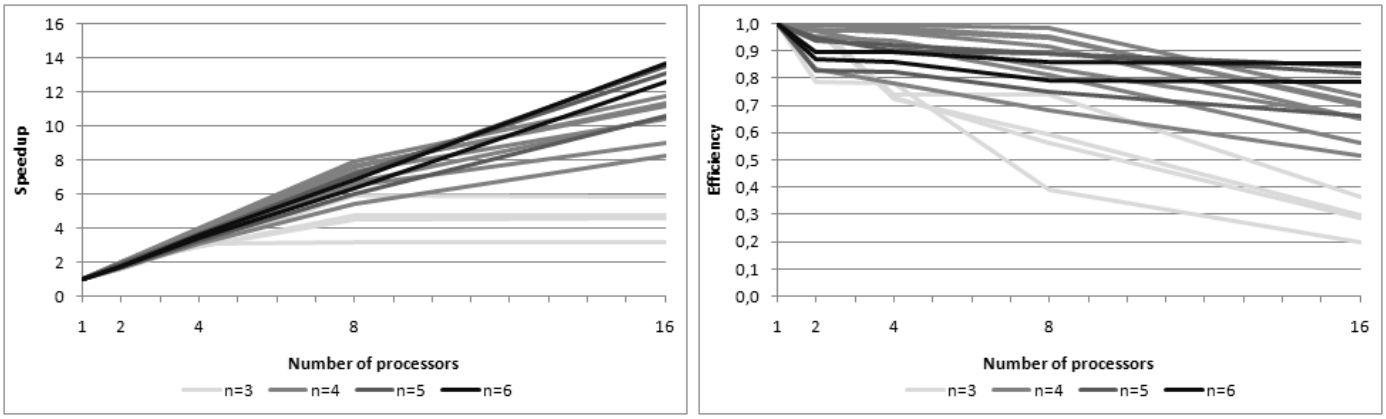
(a) (b)

Fig. 1. Speedup and efficiency of parallel version with best first selection strategy.



(a) (b)

Fig. 2. Speedup and efficiency of parallel version with statistical selection strategy.



(a) (b)

Fig. 3. Speedup and efficiency of parallel version with depth first selection strategy.

TABLE I

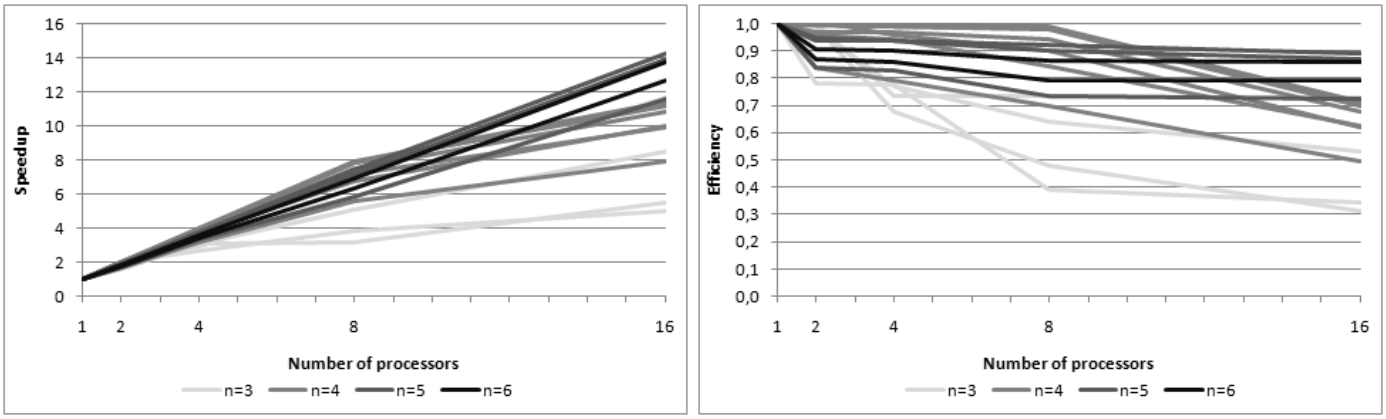
AVERAGE NUMBER OF FUNCTION EVALUATIONS AND EXECUTION TIME (S) WITH DIFFERENT SELECTION STRATEGIES

	1 p.		2 p.		4 p.		8 p.		16 p.	
n	$\overline{f.eval.}$	$\overline{t(s)}$	$\overline{f.eval.}$	$\overline{t(s)}$	$\overline{f.eval.}$	$\overline{t(s)}$	$\overline{f.eval.}$	$\overline{t(s)}$	$\overline{f.eval.}$	$\overline{t(s)}$
Best First										
3	2119769	28.22	2119810	14.18	2131476	9.08	2157114	6.57	2202417	3.52
4	1000209	36.04	1000244	18.20	1000381	9.24	1001977	4.65	1004162	3.00
5	3482357	439.45	3482357	230.56	3484261	121.24	3488265	69.22	3491818	36.76
6	4625659	3221.09	4625659	1694.01	4625659	857.17	4625659	435.98	4625659	227.64
mean	**2806998**	**931.20**	**2807018**	**489.23**	**2810444**	**249.18**	**2818254**	**129.10**	**2771356**	**67.73**
Statistical										
3	2120068	26.64	2123397	13.63	2123173	8.56	2161008	6.16	2206350	3.36
4	999775	36.29	999882	18.30	995246	9.31	1000563	4.65	969116	3.22
5	3455481	431.72	3455481	226.64	3383504	120.62	3462539	67.01	3470742	34.35
6	4625659	3220.28	4625659	1718.71	4625122	859.60	4625659	439.68	4625659	224.93
mean	**2800246**	**928.73**	**2801105**	**494.32**	**2781761**	**249.52**	**2812442**	**129.37**	**2749411**	**66.46**
Depth First										
3	2125589	24.30	2131616	13.59	2157918	8.06	2232471	5.82	2244745	5.81
4	1001147	30.68	1004290	15.82	1009851	8.00	1021786	4.19	1021763	2.81
5	3459535	436.61	3473731	247.10	3476909	125.18	3486792	66.59	3501665	36.46
6	4625659	3220.16	4625659	1804.34	4625659	899.02	4625659	470.92	4625659	236.96
mean	**2802982**	**929.44**	**2808824**	**520.21**	**2817584**	**260.06**	**2841677**	**136.88**	**2848458**	**70.51**
Breadth First										
3	2120347	22.35	2120567	12.47	2121179	7.43	2173839	5.33	2209250	3.03
4	1000431	31.28	1000597	16.11	1000643	8.12	1000729	4.21	1001150	2.93
5	3517276	450.11	3520476	253.29	3520476	127.23	3520478	68.96	3521811	35.30
6	4625659	3216.60	4625659	1781.77	4625659	892.36	4625659	465.88	4625659	234.71
mean	**2815928**	**930.08**	**2816825**	**515.91**	**2816989**	**258.79**	**2830176**	**136.09**	**2777678**	**68.99**

TABLE II

AVERAGE SPEEDUP AND EFFICIENCY OF PARALLELIZATION WITH DIFFERENT SELECTION STRATEGIES

	2 p.		4 p.		8 p.		16 p.	
n	$\overline{s_p}$	$\overline{e_p}$	$\overline{s_p}$	$\overline{e_p}$	$\overline{s_p}$	$\overline{e_p}$	$\overline{s_p}$	$\overline{e_p}$
Best First								
3	1.97	0.98	3.13	0.78	4.45	0.56	8.18	0.51
4	1.95	0.98	3.73	0.93	7.28	0.91	11.58	0.72
5	1.90	0.95	3.67	0.92	6.70	0.84	12.68	0.79
6	1.82	0.91	3.60	0.90	6.88	0.86	13.36	0.83
mean	**1.91**	**0.96**	**3.53**	**0.88**	**6.33**	**0.79**	**11.45**	**0.72**
Statistical								
3	1.93	0.96	3.18	0.80	4.61	0.58	8.07	0.50
4	1.96	0.98	3.79	0.95	7.48	0.94	11.23	0.70
5	1.92	0.96	3.64	0.91	6.85	0.86	13.13	0.82
6	1.79	0.90	3.58	0.90	6.86	0.86	13.44	0.84
mean	**1.90**	**0.95**	**3.55**	**0.89**	**6.45**	**0.81**	**11.47**	**0.72**
Depth First								
3	1.87	0.94	2.98	0.74	4.57	0.57	4.59	0.29
4	1.91	0.96	3.74	0.94	7.02	0.88	10.34	0.65
5	1.81	0.90	3.53	0.88	6.75	0.84	12.40	0.77
6	1.76	0.88	3.52	0.88	6.60	0.83	13.13	0.82
mean	**1.84**	**0.92**	**3,44**	**0.86**	**6.23**	**0.78**	**10.11**	**0.63**
Breadth First								
3	1.91	0.95	3.53	0.88	6.40	0.80	10.10	0.63
4	1.93	0.97	3.79	0.95	7.24	0.91	11.84	0.74
5	1.81	0.91	3.61	0.90	6.83	0.85	13.25	0.83
6	1.77	0.89	3.53	0.88	6.63	0.83	13.22	0.83
mean	**1.86**	**0.93**	**3.61**	**0.90**	**6.78**	**0.85**	**11.73**	**0.73**



(a) (b)

Fig. 4. Speedup and efficiency of parallel version with breadth first selection strategy.

[23] E. A. Galperin, "The cubic algorithm," *Journal of Mathematical Analysis and Applications*, vol. 112, no. 2, pp. 635–640, 1985.

[24] ——, "Precision, complexity, and computational schemes of the cubic algorithm," *Journal of Optimization Theory and Applications*, vol. 57, pp. 223–238, 1988.

[25] J. Pinter, "Extended univariate algorithms for n-dimensional global optimization," *Computing*, vol. 36, no. 1, pp. 91–103, 1986.

[26] ——, "Globally convergent methods for *n*-dimensional multiextremal optimization," *Optimization*, vol. 17, pp. 187–202, 1986.

[27] J. Pintér, "Branch-and-bound algorithms for solving global optimization problems with Lipschitzian structure," *Optimization*, vol. 19, no. 1, pp. 101–110, 1988.

[28] C. C. Meewella and D. Q. Mayne, "An algorithm for global optimization of Lipschitz continuous functions," *Journal of Optimization Theory and Applications*, vol. 57, no. 2, pp. 307–323, 1988.

[29] ——, "An efficient domain partitioning algorithms for global optimization of rational and lipschitz continuous functions," *Journal of Optimization Theory and Applications*, vol. 61, no. 2, pp. 247–270, 1989.

[30] E. Gourdin, P. Hansen, and B. Jaumard, "Global optimization of multivariate lipschitz functions: Survey and computational comparison," *Les Cahiers du GERAD*, May 1994.

[31] R. Horst, "A general class of branch-and-bound methods in global optimization with some new approaches for concave minimization," *Journal of Optimization Theory and Applications*, vol. 51, pp. 271–291, 1986.

[32] R. Horst and H. Tuy, "On the convergence of global methods in multiextremal optimization," *Journal of Optimization Theory and Applications*, vol. 54, no. 2, pp. 253–271, 1987.

[33] R. Paulavičius and J. Žilinskas, "Analysis of different norms and corresponding Lipschitz constants for global optimization in multidimensional case," *Information Technology and Control*, vol. 36, no. 4, pp. 383–387, 2007.

[34] ——, "Improved Lipschitz bounds with the first norm for function values over multidimensional simplex," *Mathematical Modelling and Analysis*, vol. 13, no. 4, pp. 553–563, 2008.

[35] ——, "Global optimization using the branch-and-bound algorithm with a combination of Lipschitz bounds over simplices," *Technological and Economic Development of Economy*, vol. 15, no. 2, pp. 310–325, 2009.

[36] R. Paulavičius, J. Žilinskas, and A. Grothey, "Parallel branch and bound for global optimization with combination of Lipschitz bounds," *Optimization Methods and Software*, no. submitted, 2010.

[37] M. Baravykaitė, R. Čiegis, and J. Žilinskas, "Template realization of generalized branch and bound algorithm," *Mathematical Modelling and Analysis*, vol. 10, no. 3, pp. 217–236, 2005.

[38] C. Jansson and O. Knüppel, "A global minimization method: The multi-dimensional case," TU Hamburg-Harburg, Tech. Rep., 1992.

[39] K. Madsen and J. Žilinskas, "Testing branch-and-bound methods for global optimization," Technical University of Denmark, Tech. Rep. IMM-REP-2000-05, 2000.