Heuristics for Longest Edge selection in Simplicial Branch and Bound

Juan F. R. Herrera[1], Leocadio G. Casado[1], Eligius M. T. Hendrix[2], and Inmaculada García[2]

[1] University of Almeria (ceiA3), Almeria, Spain
{juanfrh,leo}@ual.es
[2] Universidad de Málaga, Málaga, Spain
{eligius,igarciaf}@uma.es

Abstract. Simplicial partitions are suitable to divide a bounded area in branch and bound. In the iterative refinement process, a popular strategy is to divide simplices by their longest edge, thus avoiding needle-shaped simplices. A range of possibilities arises in higher dimensions where the number of longest edges in a simplex is greater than one. The behaviour of the search and the resulting binary search tree depend on the selected longest edge. In this work, we investigate different rules to select a longest edge and study the resulting efficiency of the branch and bound algorithm.

Keywords: bisection · branching rule · branch and bound · Global Optimization · simplices

1 Introduction

Global Optimization (GO) searches for global optima of a nonlinear function on a non-empty domain that may have local nonglobal minima. Several methods can be used to find the solution. Within deterministic methods, the branch and bound method (B&B) guarantees to find a global minimum point up to a guaranteed accuracy δ. This method iteratively divides the search space into subsets and discards those that are proven not to contain a global solution. Five rules define the method:

Branching rule It determines how to divide a problem into subproblems.
Bounding rule It defines how to obtain upper and/or lower bounds of the objective function on subproblems.
Selection rule It chooses a subproblem among all subproblems stored in a working set.
Rejection rule It discards subproblems which are proven not to contain a global solution.
Termination rule It defines when the given accuracy has been reached. Once a subproblem meets this criterion, it is not further divided. Otherwise, it is stored in the working set.

Every B&B rule plays an important role on the efficiency of the algorithm. Careless decisions in one of the rules may lead to inefficient algorithms. This work focuses on the efficiency of the branching rule using longest edge bisection within simplicial B&B optimization methods.

For some problems like mixture design, the search space is a regular simplex [5]. Here, we focus on box-constrained problems, where the search space is an n-dimensional hyper-rectangle that can be partitioned into a set of non-overlapping n-simplices. An n-simplex is a convex hull of $n+1$ affinely independent vertices. A simplex is a polyhedron in a multidimensional space, which has the minimal number of vertices. Therefore simplicial partitions are preferable in GO when the values of the objective function at all vertices of partitions are used to evaluate subregions.

A recent study shows how the number of generated sub-simplices varies when different heuristics are applied in the iterative bisection of a regular n-simplex [1] when dimensions are higher than 2. In that study, the complete binary tree is built by bisecting the heuristically-selected longest edge of a sub-simplex until the width, determined by the length of their longest edge, is smaller or equal to a given accuracy ϵ. A large reduction in the number of generated sub-simplices and therefore the size of the binary tree is achieved when one deviates from a heuristic that simply bisects the first longest edge found in terms of vertex indexation.

In this context, our initial question was about the effect when one applies different heuristics to simplicial B&B on a box constrained area, where the initial search region is not a regular simplex and the termination criterion is based on the bounding rule. In a previous study [7], we showed that the number of evaluated simplices can be reduced by not selecting the first longest edge, but that which has the smallest sum of angles with the other edges. That study generated the upper part of a binary tree running a Lipschitz B&B with a rough accuracy. We focus now on new heuristics and investigate their efficiency for a B&B algorithm that reaches at least 5% of the function range as accuracy. The question is which of the rules are most effective when the dimension of the problem is going up.

Section 2 briefly explains the main features of the used simplicial B&B algorithm. Section 3 describes the studied edge selection heuristics. The resulting search tree is compared numerically in Section 4 and Section 5 summarizes the findings.

2 Simplicial B&B Method for Multidimensional GO

We focus on the multidimensional box-constrained global optimization problem. The goal is to find at least one global minimum point x^* of

$$f^* = f(x^*) = \min_{x \in X} f(x), \tag{1}$$

where the feasible area $X \subset \mathbb{R}^n$ is a nonempty box-constrained area, i.e. it has simple upper and lower bounds for each variable. The function f is not required

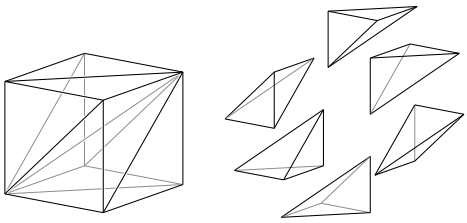


Fig. 1. Division of a hypercube into six irregular simplices

to be differentiable nor (Lipschitz) continuous. We will see in this section how one can subdivide the search space and derive simple bounds on the simplicial subsets. All ingredients are then collected into an algorithm.

Initial Space

Most B&B methods use hyper-rectangular partitions. However, other types of partitions may be more suitable for some optimization problems. Compared to the use of rectangular partitions, simplicial partitions are convenient when the feasible region is a polytope [11]. Optimization problems with linear constraints are examples where feasible regions are polytopes which can be vertex triangulated.

For the use of simplicial partitions, the feasible region is partitioned into simplices. There are two methods: over-covering and face-to-face vertex triangulation. The first strategy covers the hyper-rectangle by one simplex, that can be a regular one. The disadvantage of this method is that the search space is bigger than the feasible area and some regions can be out of the function definition. The most preferable initial covering is face-to-face vertex triangulation. It involves partitioning the feasible region into a finite number of n-dimensional simplices with vertices that are also the vertices of the feasible region. A standard method [12] is triangulation into $n!$ simplices. All simplices share the diagonal of the feasible region and have the same hyper-volume. Figure 1 depicts a hypercube of dimension three partitioned into six irregular simplices.

Bounding and Rejection Rules

Consider the objective function f with a global minimum f^* on box-constrained area X. Given a global minimum point x^*, let scalar K be such that

$$K \geq \max_{x \in X} \frac{|f(x) - f^*|}{\|x - x^*\|}, \tag{2}$$

where $\|\cdot\|$ denotes the Euclidean norm. Although this is not essential, we will work with Euclidean distance. The function $f^* + K\|x - x^*\|$ is an upper fitting

according to [2] for an arbitrary $x \in X$. Consider a set of evaluated points $x_i \in X$ with function values $f_i = f(x_i)$, then the area below

$$\varphi(x) = \max_i \{f_i - K\|x - x_i\|\} \tag{3}$$

cannot contain the global minimum (x^*, f^*). Let $f^U = \min_i f_i$ be the best function value of all evaluated points, i.e., an upper bound of f^*. Then the area $\{x \in X : \varphi(x) > f^U\}$ cannot contain the global minimum point x^*.

Now consider a simplex S with evaluated vertices $v_0, v_1, \ldots, v_n$, where $f_i = f(v_i)$. To determine the existence of optimal solution x^* in S, each evaluated vertex (v_i, f_i) provides a cutting cone:

$$\varphi_i(x) := f_i - K\|x - v_i\|. \tag{4}$$

Let Φ be defined by

$$\Phi(S) = \min_{x \in S} \max_i \varphi_i(x). \tag{5}$$

If $f^U < \Phi(S)$, then simplex S cannot contain the global minimum point x^*, and therefore S can be rejected. Notice that $\Phi(S)$ is a lower bound of f^* if S contains the minimum point x^*.

Equation (5) is not easy to determine as shown by Mladineo [10]. Therefore, alternative lower bounds of (5) can be generated in a faster way. We use two of them and take the best (highest) value.

An easy-to-evaluate case is to consider the best value of $\min_{x \in S} \varphi_i(x)$ over the vertices i. This results in a lower bound

$$\underline{\Phi}_1(S) = \max_i \{f_i - K \max_j \|v_j - v_i\|\}. \tag{6}$$

The second lower bound is based on the more elaborate analysis of infeasibility spheres in [3] and developed to non-optimality spheres in [6]. It says that S cannot contain an optimal point if it is covered completely by so-called non-optimality spheres. According to [3], if there exists a point $c \in S$ such that

$$f_i - K\|c - v_i\| > f^U \;\; i = 0, \ldots, n, \tag{7}$$

then S is completely covered and cannot contain x^*. This means that any interior point c of S provides a lower bound $\min_i\{f_i - K \max_j \|c - v_i\|\}$. Instead of trying to optimize the lower bound over c, we generate an easy-to-produce weighted average based on the radii of the spheres. Consider that $f_i > f^U$, otherwise S can contain an optimum point. Let

$$\lambda_i = \frac{K}{f_i - f^U} \tag{8}$$

and take

$$c = \frac{1}{\sum_j \lambda_j} \sum_i \lambda_i v_i. \tag{9}$$

Algorithm 1 Simplicial B&B algorithm, bisection

Require: $X,\ f,\ K, \delta$
Partition X into simplices $S_k,\ k = 1, \ldots, n!$
Start the working list as $\Lambda := \{S_k : k = 1, \ldots, n!\}$
The set of evaluated vertices $V := \{v_i \in S_k \in \Lambda\}$
Set $f^U := \min_{v \in V} f(v)$ and $x^U := \arg\min_{v \in V} f(v)$
Determine lower bounds $f_k^L = f^L(S_k)$ based on K
while $\Lambda \neq \emptyset$ **do**
  Extract a simplex $S = S_k$ from Λ
  Bisect S into $S1$ and $S2$ generating x
  if $x \notin V$ **then**
    Add x to V
    if $f(x) < f^U$ **then**
      Set $f^U := f(x)$ and $x^U := x$
      Remove all S_k from Λ with $f_k^L > f^U - \delta$
    end if
  end if
  Determine lower bounds $LB(S1)$ and $LB(S2)$
  Store $S1$ in Λ if $f^L(S1) \leq f^U - \delta$
  Store $S2$ in Λ if $f^L(S2) \leq f^U - \delta$
end while
return x^U, f^U

A second lower bound based on (7) is

$$\underline{\Phi}_2(S) = \min_i \{f_i - K\|c - v_i\|\}. \tag{10}$$

The final lower bound we consider in this paper for the B&B is the best value $f^L(S) = \max\{\underline{\Phi}_1(S), \underline{\Phi}_2(S)\}$.

Selection and Termination Rules

The algorithm performs a depth-first search by selecting the sub-simplex with the smallest $f^L(S)$ value among those generated in the last division, until the final accuracy is reached or both new sub-simplices are rejected. In general, depth-first search minimizes the memory requirement of the algorithm. A simplex S is discarded when $f^L(S) + \delta > f^U$ for an accuracy $\delta > 0$. The steps of the B&B algorithm are described in Algorithm 1.

3 Longest Edge Bisection

Literature discusses many methods to subdivide a simplex [8]. One of them is the Longest Edge Bisection (LEB), which is a popular way of iterative division in the finite element method, since it is very simple and can easily be applied in higher dimensions [4]. This method consists of splitting a simplex using the hyperplane that connects the middle point of the longest edge of a simplex with

the opposite vertices. This is illustrated in Figure 2 that also shows that in higher dimensions there can be several longest edges. For our study, we should notice that due to the initial partition as sketched in Figure 1, for $n = 3$ the longest edge is unique in all generated subsets. This means that to observe what happens with a choice of the longest edge to the search tree, we should focus on dimensions higher than 3. We formulate several rules to select the longest edge.

Fig. 2. Longest Edge Bisection generating a sub-simplex with three longest edges

The most common edge selection rule in LEB is the following:

LEB$_1$ Natural coding implicitly selects a longest edge being the first one found. The sequence depends on the coding and storing of the vertices and edges, i.e. the index number assigned to each vertex of the simplex. When a simplex is split into two new sub-simplices, the new vertex of each sub-simplex has the same index as the one it substitutes.

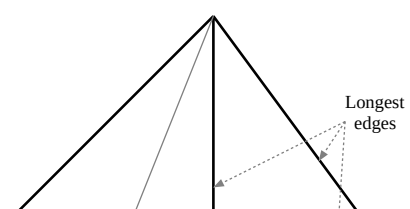
Our preliminary studies show the existence of many sub-simplices having more than one longest edge when LEB is used as iterative partition rule in a simplicial B&B algorithm.

In order to reduce the search tree size, other heuristics for selecting the longest edge in the division of a regular n-simplex are investigated to be used to simplicial B&B algorithms. They are summarized below:

LEB$_\alpha$ For each vertex in a longest edge, the sum of the angles between edges ending at that vertex is determined and the longest edge corresponding to the smallest sum is selected.

LEB$_C$ Bisects the longest edge with the largest distance from its middle point to the centroid of the simplex.

LEB$_M$ Determines the distance from a longest edge midpoint to the other vertices. It then selects that longest edge that has the maximum sum of distances to the other vertices.



LEB$_W$ Selects an edge that has not been involved in many bisections yet via a weight system. The initial set of evaluated vertices (line 3 of Algorithm 1) are set to $w_i := 0$. A new vertex v_i (generated by the branching rule, line 8 of Algorithm 1) is initiated with weight $w_i := 1$. Each time vertex v_i belongs to a divided edge, its weight is updated to $w_i := w_i + 1 \mod n$. For each longest edge defined by vertices (v_i, v_j), the two weights (w_i, w_j) are summed and the one with smallest sum is selected.

The research goal is to determine a LEB rule that minimizes the search tree generated by a simplicial B&B algorithm measured as the total number of generated nodes (subsimplices).

4 Comparison of the Selection Rules

A set of several test functions has been built to measure the tree generated by the set of LEB strategies discussed in the previous section. A complete suite of test functions can be found in [9]. From this set, we select a subset of functions that allow varying the dimension of the problem. We remind the reader that the often-used low dimensional instances are not appropriate to measure the difference of the generated tree as for dimensions $n \leq 3$ there is no choice on the selected longest edge to be bisected.

For each test function, at least one global minimum point is known and we determined the sharpest value of parameter K in (2) that we could find using a multistart approach. For instances like *MaxMod* or *Zakharov*, a value for K can be determined analytically. The data of the corresponding test-bed is given in Table 1. A description of the test instances with the considered minimum point and the function range $[f^*, \overline{f}]$ on the given domain is provided in the appendix, where $\overline{f}$ is the maximum function value on the domain. The depth of the generated B&B tree is mainly determined by the accuracy δ. To obtain reasonable size trees, in the experiments the value of the accuracy δ is set on $\delta = 0.05\,(\overline{f} - f^*)$.

Table 2 shows the numerical results for a search domain defined in a four-dimensional space. The computational effort is captured in terms of the number of generated and evaluated simplices in the corresponding B&B tree, see lines 5 and 16 of Algorithm 1. Column LEB$_1$ denotes the total number of evaluated simplices and the other columns provide the reduction with respect to LEB$_1$ generated by the remaining rules, expressed as a percentage. Rules LEB$_\alpha$ and LEB$_M$ provide higher reductions than LEB$_C$ and LEB$_W$, which in some cases perform more simplex evaluations than LEB$_1$. The interesting aspect is that selection rule LEB$_M$ is easier to generate than selection rule LEB$_\alpha$ in terms of computational operations.

A side result not related to our original question is the effectiveness of the bounding rule. We measured that the more sophisticated bound $\underline{\Phi}_2$ is lower for 75% of the evaluated sub-simplices than the simpler lower bound $\underline{\Phi}_1$.

Table 3 contains the results for dimension $n = 5$. In this case, LEB$_M$ provides a higher reduction than the rest of rules. LEB$_\alpha$ shows reductions similar to LEB$_C$

Table 1. Test instances for dimension $n = 4, 5, 6$, and the corresponding K_n values

No.	Test problem	Domain	K_4	K_5	K_6
1	Ackley	$[-30, 30]^n$	5.4	4.9	4.4
2	Dixon & Price	$[-10, 10]^n$	21,646.5	29,086.1	37,248.8
3	Holzman	$[-10, 10]^n$	5,196.2	7,000.0	9,000.0
4	MaxMod	$[-10, 10]^n$	1.0	1.0	1.0
5	Perm	$[-n, n]^n$	183,998.1	31,159,684.8	7,746,536,437.2
6	Pinter	$[-10, 10]^n$	60.0	75.3	88.7
7	Quintic	$[-10, 10]^n$	29,712.0	33,219.0	36,389.7
8	Rastrigin	$[-5.12, 5.12]^n$	91.8	102.7	112.5
9	Rosenbrock	$[-5, 10]^n$	168,005.3	190,517.7	210,669.4
10	Schwefel 1.2	$[-10, 10]^n$	176.8	176.8	444.6
11	Zakharov	$[-5, 10]^n$	312,645.0	1,415,285.7	4,962,758.1

Table 2. Experimental results for $n = 4$, number of evaluated simplices by LEB_1 and the reduction by the other rules

No.	Test problem	LEB_1	LEB_α	LEB_C	LEB_M	LEB_W
1	Ackley	8,467,608	35%	−3%	35%	0%
2	Dixon & Price	2,312,132	28%	8%	28%	5%
3	Holzman	3,419,030	25%	4%	25%	2%
4	MaxMod	5,742,672	54%	9%	54%	5%
5	Perm	39,987,438	3%	2%	31%	0%
6	Pinter	4,896,640	26%	−2%	26%	−1%
7	Quintic	2,527,376	29%	8%	29%	6%
8	Rastrigin	117,620,808	12%	0%	13%	1%
9	Rosenbrock	2,806,400	27%	0%	27%	−1%
10	Schwefel 1.2	1,857,322	30%	6%	30%	3%
11	Zakharov	5,299,018	30%	−6%	30%	−3%

and LEB_W, in contrast with dimension $n = 4$ reported in Table 2. Interesting is also the contrast with earlier findings where the rules were applied for refining a unit simplex in [1]. There the LEB_α rule clearly dominates other rules.

Table 4 provides numerical results for a subset of the test instances that could be run within a reasonable computation time for dimension $n = 6$. One can observe that LEB_M provides a larger reduction than the other rules.

In all of the experiments we found that the longest edge is not unique in about 70% of the sub-simplices providing the opportunity for a selection rule. Moreover, heuristics LEB_α and LEB_W sometimes provide not unique criterion values to the multiple longest edges. This calls for a second criterion to be evaluated. Interesting enough, it appeared that additional criteria lead to worse reductions, such that one best takes the first edge with the smallest sum of angles or smallest sum of weights for heuristics LEB_α and LEB_W respectively.

Table 3. Experimental results for $n = 5$, number of evaluated simplices by LEB_1 and the reduction by the other rules

No.	Test problem	LEB_1	LEB_α	LEB_C	LEB_M	LEB_W
1	Ackley	1,010,945,400	1%	6%	26%	7%
2	Dixon & Price	123,575,850	9%	14%	25%	13%
3	Holzman	219,996,634	10%	12%	24%	11%
4	MaxMod	1,877,094,680	12%	7%	20%	3%
5	Perm	166,831,502	13%	5%	19%	3%
6	Pinter	989,052,844	10%	5%	20%	4%
7	Quintic	261,009,818	9%	13%	25%	14%
8	Rastrigin	23,085,565,464	4%	7%	18%	8%
9	Rosenbrock	175,613,436	0%	7%	22%	7%
10	Schwefel 1.2	87,628,502	12%	13%	25%	10%
11	Zakharov	603,678,276	0%	3%	18%	5%

Table 4. Experimental results for $n = 6$, number of evaluated simplices by LEB_1 and the reduction by the other rules

No.	Test problem	LEB_1	LEB_α	LEB_C	LEB_M	LEB_W
2	Dixon & Price	8,203,852,060	9%	14%	24%	13%
3	Holzman	16,628,924,978	17%	13%	26%	6%
7	Quintic	24,424,636,700	16%	12%	26%	4%
9	Rosenbrock	11,689,814,082	10%	9%	24%	6%
10	Schwefel 1.2	5,154,080,906	20%	16%	37%	20%

5 Conclusion

The question in this paper is how different selection rules for selecting the appropriate longest edge in simplicial B&B algorithms may influence the size of the generated search tree. To investigate this question, a simplicial B&B algorithm is used where non-optimal area is cut away via the concept of an upper fitting. Evaluating five LEB heuristics on a set of test instances in dimensions $n = 4, 5, 6$, shows that a rule called LEB_M gives the best performance. The search tree can be reduced up to about 25% compared to the rule that selects the first longest edge as stored in an implementation of the simplicial B&B.

Acknowledgments. This work has been funded by grants from the Spanish Ministry (TIN2012-37483) and Junta de Andalucía (P11-TIC-7176 and P12-TIC-301), in part financed by the European Regional Development Fund (ERDF). J.F.R. Herrera is a fellow of the Spanish FPU program.

References

1. Aparicio, G., Casado, L.G., Hendrix, E.M.T., Garcia, I., Toth, B.G.: On computational aspects of a regular n-simplex bisection. In: P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on. pp. 513–518 (2013)
2. Baritompa, W.: Customizing methods for global optimization, a geometric viewpoint. Journal of Global Optimization 3(2), 193–212 (1993)
3. Casado, L.G., Hendrix, E.M., García, I.: Infeasibility spheres for finding robust solutions of blending problems with quadratic constraints. Journal of Global Optimization 39(4), 577–593 (2007)
4. Hannukainen, A., Korotov, S., Kek, M.: On numerical regularity of the face-to-face longest-edge bisection algorithm for tetrahedral partitions. Science of Computer Programming 90, 34 – 41 (2014)
5. Hendrix, E.M.T., Casado, L.G., García, I.: The semi-continuous quadratic mixture design problem: Description and branch-and-bound approach. Eur. J. Oper. Res. 191(3), 803–815 (2008)
6. Hendrix, E.M.T., Casado, L.G., Amaral, P.: Global optimization simplex bisection revisited based on considerations by Reiner Horst. In: Murgante, B., et al. (eds.) Computational Science and Its Applications – ICCSA 2012, Lecture Notes in Computer Science, vol. 7335, pp. 159–173. Springer (2012)
7. Herrera, J.F.R., Casado, L.G., Hendrix, E.M.T., García, I.: On simplicial longest edge bisection in Lipschitz global optimization. In: Murgante, B., et al. (eds.) Computational Science and Its Applications – ICCSA 2014, Lecture Notes in Computer Science, vol. 8580, pp. 104–114. Springer (2014)
8. Horst, R., Tuy, H.: Global Optimization (Deterministic Approaches). Springer, Berlin (1990)
9. Jamil, M., Yang, X.: A literature survey of benchmark functions for global optimization problems. Int. Journal of Mathematical Modelling and Numerical Optimisation 4(2), 150–194 (2013)
10. Mladineo, R.H.: An algorithm for finding the global maximum of a multimodal multivariate function. Math. Program. 34, 188–200 (1986)
11. Paulavičius, R., Žilinskas, J.: Simplicial Global Optimization. SpringerBriefs in Optimization, Springer New York (2014)
12. Todd, M.J.: The computation of fixed points and applications, Lecture Notes in Economics and Mathematical Systems, vol. 24. Springer-Verlag (1976)

Appendix: Function Definitions

The test function description is given with the considered minimum point x^*, minimum value f^* and maximum value $\overline{f}$ over the domain in Table 1 for dimensions $n = 4, 5, 6$.

Ackley

$$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)\right) + 20 + e$$

$x^* = 0,\ f^* = 0,\ \overline{f}_4 = 22.2,\ \overline{f}_5 = 22.2,\ \overline{f}_6 = 22.2$

Dixon & Price

$$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{D} i(2x_i^2 - x_{i-1})^2$$

$x_i^* = 2^{\frac{2-2^i}{2^i}},\ f^* = 0,\ \overline{f}_4 = 397{,}021,\ \overline{f}_5 = 617{,}521,\ \overline{f}_6 = 88{,}212$

Holzman

$$f(x) = \sum_{i=1}^{n} i x_i^4$$

$x^* = 0,\ f^* = 0,\ \overline{f}_4 = 100{,}000,\ \overline{f}_5 = 150{,}000,\ \overline{f}_6 = 210{,}000$

MaxMod

$$f(x) = \max(|x_i|)$$

$x^* = 0,\ f^* = 0,\ \overline{f}_4 = 10,\ \overline{f}_5 = 10,\ \overline{f}_6 = 10$

Perm

$$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} (j^i + \beta) \left(\left(\frac{x_j}{j} \right)^i - 1 \right) \right)^2$$

$x_i^* = i,\ f^* = 0,\ \overline{f}_4 = 809{,}249,\ \overline{f}_5 = 476{,}712{,}082,\ \overline{f}_6 = 59{,}926{,}724{,}566$

Pinter

$$f(x) = \sum_{i=1}^{n} i x_i^2 + \sum_{i=1}^{n} 20i \sin^2 A + \sum_{i=1}^{n} i \log 10(1 + iB^2)$$

where

$$\begin{cases} A = (x_{i-1} \sin x_i + \sin x_{i+1}) \\ B = (x_{i-1}^2 - 2x_i + 3x_{i+1} - \cos x_i + 1) \end{cases}$$

where $x_0 = x_n$ and $x_{n+1} = x_0$.
$x^* = 0,\ f^* = 0,\ \overline{f}_4 = 500,\ \overline{f}_5 = 625,\ \overline{f}_6 = 751$

Quintic

$$f(x) = \sum_{i=1}^{n} |x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i - 4|$$

$x_i^* = -1,\ f^* = 0,\ \overline{f}_4 = 532{,}816,\ \overline{f}_5 = 668{,}520,\ \overline{f}_6 = 802{,}224$

Rastrigin

$$f(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i))$$

$x^* = 0,\ f^* = 0,\ \overline{f}_4 = 153,\ \overline{f}_5 = 190,\ \overline{f}_6 = 231$

Rosenbrock

$$f(x) = \sum_{i=1}^{n-1}[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2]$$

$x_i^* = 1,\ f^* = 0,\ \overline{f}_4 = 2{,}722{,}743,\ \overline{f}_5 = 3{,}532{,}824,\ \overline{f}_6 = 4{,}342{,}905$

Schwefel 1.2

$$f(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_i\right)^2$$

$x^* = 0,\ f^* = 0,\ \overline{f}_4 = 3{,}000,\ \overline{f}_5 = 5{,}500,\ \overline{f}_6 = 9{,}100$

Zakharov

$$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{n} i x_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{n} i x_i\right)^4$$

$x^* = 0,\ f^* = 0,\ \overline{f}_4 = 6{,}252{,}900,\ \overline{f}_5 = 31{,}646{,}750,\ \overline{f}_6 = 121{,}562{,}250$