Chapter 2
Lipschitz Optimization with Different Bounds over Simplices

2.1 Lipschitz Optimization

Lipschitz optimization is one of the most deeply investigated subjects of global optimization. It is based on the assumption that the real-valued objective function $f(\mathbf{x})$ and sometimes its first derivative $f'(\mathbf{x})$ are Lipschitz-continuous functions on $\mathbb{D}$, i.e.,

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L \|\mathbf{x} - \mathbf{y}\|, \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{D}, \quad 0 < L < \infty, \tag{2.1}$$

or

$$\left|f'(\mathbf{x}) - f'(\mathbf{y})\right| \leq K \|\mathbf{x} - \mathbf{y}\|, \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{D}, \quad 0 < K < \infty, \tag{2.2}$$

where $f'(\mathbf{x})$, $f'(\mathbf{y})$ are the directional derivatives calculated at the points $\mathbf{x}, \mathbf{y}$ in the direction $y - x$, L, K are Lipschitz constants, $\mathbb{D} \subset \mathbb{R}^n$ is compact, and $\|\cdot\|$ denotes a norm.

The advantages and disadvantages of Lipschitz global optimization methods are discussed in [49,55,110,134]. These methods are deterministic and provide the same result each time unlike stochastic methods; there is no need to run the algorithm multiple times. In contrast to heuristic methods there is no need for parameter tuning. The algorithms can provide bounds on how far they are from the optimum function value, and hence can use stopping criteria that are more meaningful than a simple iteration limit. The methods guarantee to find an approximation of the solution to a specified accuracy within finite time, as opposed to direct search methods or stochastic methods. They may be used in situation when an analytical description of the objective function is not available (but an estimate of Lipschitz constant is known), as opposed to interval optimization methods.

The main disadvantage of Lipschitz methods is the requirement to provide the Lipschitz constant of the objective function. One of the main questions to be considered in Lipschitz optimization is: how can the Lipschitz constant be obtained?

R. Paulavičius and J. Žilinskas, *Simplicial Global Optimization*,
SpringerBriefs in Optimization, DOI 10.1007/978-1-4614-9093-7_2,

There are at least four approaches to obtain an estimate of the Lipschitz constant L [125]:

1. It can be given a priori [3, 59, 89, 112]. This case is very important from the theoretical viewpoint but is not frequently encountered in practice.
2. Its adaptive global estimate over the whole domain can be used [59,71,109,134].
3. Local Lipschitz constants can be estimated adaptively [70,79,116,117,124,129, 134];
4. Its estimates can be chosen from a set of possible values [34,64,123,124].

Similar to the Lipschitz constant of the objective function, there are several ways to estimate the Lipschitz constant K of the first derivative. There exist algorithms using an a priori given estimate of K [9, 119], its adaptive estimates [41, 119], adaptive estimates of local Lipschitz constants [27, 44, 118–120], and algorithms working with a number of Lipschitz constants chosen from a set of possible values varying from zero to infinity [72,73].

There also exist techniques working with Lipschitz multi-extremal constraints [121,127,128,134].

The Lipschitz optimization algorithms investigated in this book can be assigned to the first, second, and fourth approaches to obtain estimate of L. In this chapter we assume that the Lipschitz constant L is known in advance.

The value of the Lipschitz constant in inequality (2.1) depends on the used q-norm

$$\begin{cases} \|\mathbf{x}\|_q = \left(\sum_{i=1}^{n} |x_i|^q\right)^{1/q}, & 1 \le q < \infty, \mathbf{x} \in \mathbb{R}^n, \\ \|\mathbf{x}\|_\infty = \max\{|x_1|, |x_2|, \ldots, |x_n|\}. \end{cases}$$

In Lipschitz optimization the Euclidean norm ($q = 2$) is used most often, but other norms can also be considered [96, 97]. Let us formulate a proposition in which we present the dependence between the Lipschitz constant and the used norm.

Proposition 2.1. *Let $\mathbb{D} \subset \mathbb{R}^n$ be a convex bounded closed set, and let $f(\mathbf{x}) : \mathbb{D} \to \mathbb{R}$ be a continuously differentiable function on an open set containing $\mathbb{D}$. Then $f(\mathbf{x})$ is a Lipschitz function and for $\forall \mathbf{x}, \mathbf{y} \in \mathbb{D}$ the following inequality holds:*

$$|f(\mathbf{x}) - f(\mathbf{y})| \le L_p \|\mathbf{x} - \mathbf{y}\|_q, \tag{2.3}$$

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

Proof. For every pair $\mathbf{x}, \mathbf{y} \in \mathbb{D}$, there exists $\mathbf{z} = \mathbf{x} + \lambda(\mathbf{y} - \mathbf{x}) \in \mathbb{D}$, $0 < \lambda < 1$, such that

$$|f(\mathbf{y}) - f(\mathbf{x})| = |\nabla f(\mathbf{z}) \cdot (\mathbf{y} - \mathbf{x})|,$$

(by the mean-value theorem)

$$|f(\mathbf{y}) - f(\mathbf{x})| \le \sum_{i=1}^{n} \left| \frac{\partial f}{\partial z_i}(y_i - x_i) \right| \le \left(\sum_{i=1}^{n} \left| \frac{\partial f}{\partial z_i} \right|^p \right)^{1/p} \left(\sum_{i=1}^{n} |y_i - x_i|^q \right)^{1/q},$$

where $1/p + 1/q = 1$, $1 \le p, q \le \infty$ (using Hölder's inequality).
Since

$$\|\nabla f(\mathbf{z})\|_p = \left(\sum_{i=1}^{n} \left| \frac{\partial f}{\partial z_i} \right|^p \right)^{1/p}, \quad \|\mathbf{x} - \mathbf{y}\|_q = \left(\sum_{i=1}^{n} |x_i - y_i|^q \right)^{1/q},$$

therefore

$$|f(\mathbf{x}) - f(\mathbf{y})| \le \|\nabla f(\mathbf{z})\|_p \|\mathbf{x} - \mathbf{y}\|_q \le \max \left\{ \|\nabla f(\mathbf{z})\|_p : \mathbf{z} \in \mathbb{D} \right\} \|\mathbf{x} - \mathbf{y}\|_q.$$

Denoting $L_p = \max \left\{ \|\nabla f(\mathbf{z})\|_p : \mathbf{z} \in \mathbb{D} \right\}$ we get (2.3).

In conclusion, the supremum of the p norm of the gradient of the function is the Lipschitz constant for the q norm in the Lipschitz condition.

In this chapter we assume that the Lipschitz constants are known. For the experimental investigation of the Lipschitz bounds and algorithms, the constants have been estimated following the procedure described in [49]. A grid search algorithm of 1000^n points for $(n = 2, 3)$ and 100^n points for $(n = 4, 5, 6)$ dimensional test problems (which are given together with their derivatives in Appendix A) has been used and the obtained estimates are thus close to the actual Lipschitz constants.

The estimated Lipschitz constants' values $L_p : p = 1, 2, \infty$ and the optimal vectors $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{D}$ where the maximal p-norm was obtained are shown in Tables 2.1 and 2.2. In the tables the symbol # means "Test Problem number."

The estimates of the Euclidean Lipschitz constants L_2 were compared to those given in [49] which are repeated in Table 2.1. Most of the estimates coincide. However, the estimates of the Lipschitz constants of the test problems 14 and 15 are quite different. We believe that our estimates are more correct. In the case of the test problem 14, the values of derivatives at the feasible point $(1, 1, 0) \in \mathbb{D} = [0, 1]^3$ are equal to 200, and therefore from (2.3)

$$L_2 = \|\nabla f(1, 1, 0)\|_2 = \sqrt{200^2 + 200^2 + 200^2} \approx 346.41.$$

2.2 Classical Lipschitz Bounds

The most studied case of problem (1.1) is the bound constrained univariate one ($n = 1$), for which numerous algorithms have been proposed, compared, and theoretically investigated. An excellent comprehensive survey is contained in [49]. In this book, we are mainly interested in the multivariate case ($n \ge 2$).

Table 2.1 Lipschitz constants values for $(n = 2, 3)$ dimensional test problems using $\infty, 2, 1$-norms

	Lipschitz constants				Optimal vector		
#	L_1	L_2	L_2[49]	L_∞	z_1	z_2	z_3
1	50.27	50.27	50.2665	50.27	1.000	1.000	–
2	7.98	6.32	6.3183	6.00	1.000	0.380	–
3	141.34	112.44	112.44	107.09	0.000	0.000	–
4	2.00	2.00	2	2.00	1.000	1.000	–
5	2.00	1.42	$\sqrt{2}$	1.00	0.000	0.000	–
6	1284.80	1059.59	1059.59		4.000	4.000	–
				1034.00	−2.000	4.000	–
7	14708.00	12781.70	12781.7	12608.00	−3.000	−1.500	–
8	122.00	86.32	86.3134	63.00	−2.500	4.500	–
9	2639040.00	2225890.00	2225892	2177520.00	−2.000	2.000	–
10	24.00	17.03	17.034	13.04	4.000	−3.000	–
11	64.14	47.55	47.426	42.16	1.000	2.000	–
12	80.40	56.85	56.852	40.40	1.000	1.000	–
13		993.72	988.82	993.72	0.010	0.458	–
	995.29				0.010	0.020	–
14	600.00	346.41	244.95	200.00	1.000	1.000	0.000
15		18.33	42.626	18.32	0.091	0.556	0.990
	21.36				0.202	0.444	0.970
16		988.79	971.59	988.79	0.010	0.130	0.250
	991.05				0.010	0.020	0.210
17	33.52	19.39	19.39		4.000	3.528	4.000
				14.80	1.177	4.000	4.000
18	4.77	2.92	2.919	2.38	1.000	1.000	1.000
19	224.00	130.00	130.0	80.00	2.000	−2.000	2.000
20	33616.00	22267.90	–	16808.00	−3.000	−3.000	−3.000

In Lipschitz optimization the lower bound for $f(\mathbf{x})$ over a subregion $\mathbb{I} \subseteq \mathbb{D}$ is evaluated by exploiting the Lipschitz condition (2.3). It follows from (2.3) that for all $\mathbf{x}, \mathbf{y} \in \mathbb{I}$

$$f(\mathbf{x}) \geq f(\mathbf{y}) - L_p \|\mathbf{x} - \mathbf{y}\|_q.$$

If $\mathbf{y} \in \mathbb{I}$ is fixed, then the concave function

$$F_{\mathbf{y}}^q(\mathbf{x}) = f(\mathbf{y}) - L_p \|\mathbf{x} - \mathbf{y}\|_q \tag{2.4}$$

underestimates $f(\mathbf{x})$ over $\mathbb{I}$. The superscript q indicates the norm used for the bound calculation.

Thus an algorithm for Lipschitz optimization may be built. Start from an arbitrary point $\mathbf{x}_1 \in \mathbb{I}$ and define the first underestimating function by

$$F_1^q(\mathbf{x}) = f(\mathbf{x}_1) - L_p \|\mathbf{x} - \mathbf{x}_1\|_q. \tag{2.5}$$

Table 2.2 Lipschitz constants values for ($n \geq 4$) test functions using $\infty, 2, 1$-norms

	Lipschitz constants			Optimal vector					
#	L_1	L_2	L_∞	z_1	z_2	z_3	z_4	z_5	z_6
21	1370.2	1196.4	1195.5	−10.00	−10.00	−10.00	−9.58	–	–
22	108720	60402	36026	4.00	−4.00	−4.00	−4.00	–	–
23	204.1	102.4	56.1	10.00	0.00	0.00	0.00	–	–
24	300.1	151.5	86.1	0.00	0.00	10.00	0.00	–	–
25	408.2	204.5	110.8	0.00	0.00	10.00	0.00	–	–
26	600.0	313.7	200.0	10.00	10.00	10.00	10.00	–	–
27	92216	48252		−4.00	−4.00	5.00	5.00	–	–
			29270	5.00	5.00	−4.00	−4.00	–	–
28	26.1	14.4	8.3	−10.00	−10.00	−10.00	−10.00	–	–
29		370.7	369.7	−5.00	−5.00	−5.00	−5.00	−5.00	–
	476.7			−5.00	−5.00	−5.00	−4.92	−4.58	–
30	264385	129425	66032	5.00	−5.00	−5.00	−5.00	−5.00	–
31	34.4	16.6	8.3	−10.00	−10.00	−10.00	−10.00	−10.00	–
32		370.9	369.7	−5.00	−5.00	−5.00	−5.00	−5.00	–4.59
	488.7			−5.00	−5.00	−5.00	−5.00	−4.92	–4.58
33	546547	240918	109238	6.00	−6.00	−6.00	−6.00	−6.00	–6.00

Then by choosing the "most promising" point $\mathbf{x}_2 \in \mathbb{I}$ (with the minimal lower-bounding function $F_1^q(\mathbf{x})$ value)

$$\mathbf{x}_2 = \arg\min_{\mathbf{x}\in\mathbb{I}} F_1^q(\mathbf{x}),$$

we obtain another concave underestimating function $f(\mathbf{x}_2) - L_p\|\mathbf{x} - \mathbf{x}_2\|_q$, and the best underestimator of $f(\mathbf{x})$ on $\mathbb{I}$, given the information obtained so far, is

$$F_2^q(\mathbf{x}) = \max_{i=1,2}\{f(\mathbf{x}_i) - L_p\|\mathbf{x} - \mathbf{x}_i\|_q\}.$$

By analogy, in step k the underestimating function $F_k^q(\mathbf{x})$ is

$$F_k^q(\mathbf{x}) = \max_{i=1,\ldots,k}\left\{f(\mathbf{x}_i) - L_p\|\mathbf{x} - \mathbf{x}_i\|_q\right\}, \tag{2.6}$$

and the next iteration point $\mathbf{x}_{k+1}$ is given by

$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}\in\mathbb{I}} F_k^q(\mathbf{x}). \tag{2.7}$$

The lower bound based on (2.6) and (2.7) calculations will be referred as the φ type (or Piyavskii [112] type) bound

$$\varphi^q(\mathbb{I}) = \min_{\mathbf{x}\in\mathbb{I}} \max_{\mathbf{x}_i\in\mathbb{T}} F_{\mathbf{x}_i}^q(\mathbf{x}), \tag{2.8}$$

where $\mathbb{T}$ is a finite set of distinct points in $\mathbb{I}$.

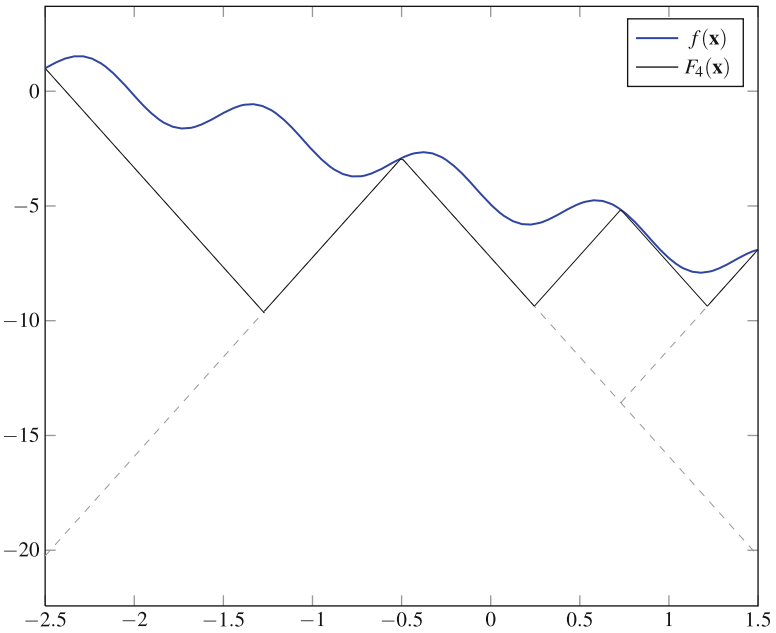

Fig. 2.1 Visualization of Piyavskii type lower bounding function $F_k(\mathbf{x})$ in univariate case

In the univariate case ($n = 1$) all norms are equal, therefore q-norm superscript is not used. The function $\max_{\mathbf{x}_i \in \mathbb{T}} F_{\mathbf{x}_i}(\mathbf{x})$ is piecewise linear (see Fig. 2.1), and φ can be determined in a simple straightforward way [49]. Therefore, many univariate algorithms use the bound φ, where the set $\mathbb{T}$ is suitably updated in an iterative way. The most often studied of these methods is due to Piyavskii [111, 112]. It was independently rediscovered by Shubert [130]. Figure 2.1 visualizes univariate Piyavskii method situation after $k = 4$ steps, where

$$f(x) = -\frac{13}{6}x + \sin\left(\frac{13}{4}(2x+5)\right) - \frac{53}{12}, \quad \mathbb{D} = [-2.5, 1.5], \quad L = 8.66. \tag{2.9}$$

When $\mathbb{I}$ is a two-dimensional rectangle in $\mathbb{R}^2$, φ type bound with the Euclidean norm ($q = 2$) can still be evaluated by geometric arguments which take into account the conical shape of the lower bounding function [89]. For ($n > 2$), however, problem (2.8) constitutes a difficult optimization problem. Convergent deterministic methods for solving the multivariate unconstrained problem (1.1) fall into three main classes.

First, multivariate Lipschitz optimization can be reduced to the univariate case. Following this idea, a nested optimization scheme was proposed by Piyavskii [111, 112]. Another scheme based on filling the feasible space by Peano curve was studied by Butz [11] and Strongin [132].

The second class contains direct extensions of Piyavskii's method [111, 112] to the multivariate case. Various modifications using the Euclidean norm: Piyavskii [112], Mladineo [89, 90], Jaumard et al. [61], or other norms or close approximations: Mayne and Polak [84], Wood [141, 142], Zhang et al. [144] have been proposed. Note that when the Euclidean norm is used in the multivariate case, the lower bounding functions are envelopes of circular cones with parallel symmetry axes. A problem of finding the minimum (2.8) of such a bounding function becomes a difficult global optimization problem involving a system of quadratic and linear equations. Most of these approaches are quite ingenious from a theoretical viewpoint, but the inherent difficulty of subproblems has limited practical applicability to dimension $n > 2$ in general unconstrained problems. For an excellent survey and numerical tests, see [49]. Most of these algorithms can be improved by interpreting them as branch-and-bound methods.

The third class contains many simplicial and rectangular branch-and-bound techniques, but, in general, considerably weaker bounds: Galperin [37, 38], Pinter [106–108], Meewella and Mayne [86, 87], Gourdin et al. [45]. These algorithms fit into the general framework proposed by Horst [53], Horst and Tuy [58, 59]. These algorithms differ in the selection rules, the ways how branching is performed and bounds are computed.

Lipschitz branch-and-bound algorithms usually use considerably weaker bounds. Most often, weaker bounds belong to the following two simple families μ_1 and μ_2. Let

$$\delta_q(\mathbb{I}) = \max\left\{\|\mathbf{x}-\mathbf{y}\|_q : \mathbf{x}, \mathbf{y} \in \mathbb{I}\right\}$$

denote the diameter of $\mathbb{I}$. For example, if $\mathbb{I} = \{\mathbf{x} \in \mathbb{R}^n : a_i \le x_i \le b_i\}$ is an n-rectangle, then $\delta_q(\mathbb{I}) = \|\mathbf{b}-\mathbf{a}\|_q$, and if $\mathbb{I}$ is an n-simplex, then the diameter $\delta_q(\mathbb{I})$ is the length of its longest edge.

A simpler lower bound can be derived from (2.4):

$$\mu_1^q(\mathbb{I}) = \max_{\mathbf{y}\in\mathbb{T}} f(\mathbf{y}) - L_p\delta_q(\mathbb{I}), \tag{2.10}$$

where $\mathbb{T} \subset \mathbb{I}$ is a finite sample of points in $\mathbb{I}$, where the function values of f have been evaluated. If $\mathbb{I}$ is a rectangle or a simplex, the set $\mathbb{T}$ often coincides with (or is a subset of) the vertex set $\mathbb{V}(\mathbb{I})$. Note, however, that it might be worthwhile to take into account interior points. The middle point $\mathbf{m} = (\mathbf{a}+\mathbf{b})/2$ of an n-rectangle $\mathbf{a} \le \mathbf{x} \le \mathbf{b}$ yields the bound

$$\mu_m^q(\mathbb{I}) = f(\mathbf{m}) - L_p\delta_q(\mathbb{I})/2.$$

A tighter but computationally more expensive than (2.10) bound is

$$\mu_2^q(\mathbb{I}) = \max_{\mathbf{y}\in\mathbb{T}} \left\{ f(\mathbf{y}) - L_p \max_{\mathbf{z}\in\mathbb{V}(\mathbb{I})} \|\mathbf{y}-\mathbf{z}\|_q \right\}. \tag{2.11}$$

This situation suggests that searching for new lower bounds, which are tighter than (2.10) and (2.11), but computationally less expensive than (2.8), might be worthwhile. In the next sections, lower bounds over hyper-rectangles and simplices are discussed. These improved bounds are stronger than (2.10) and (2.11), but they are usually still computationally cheap.

2.3 Impact of Norms on Lipschitz Bounds

In the one-dimensional case, all norms are equal. But in the multidimensional case evaluated bounds depend on the used norm [96, 97]. In this section, we investigate how the Lipschitz bounds are influenced by the used norms and the corresponding Lipschitz constants when μ_2^q bound (2.11) is used.

Let us investigate simplicial partitioning of a hyper-cubic feasible region. In two-dimensional case such a feasible region $\mathbb{D} = [0,1]^2$ is a unit square, which by face-to-face vertex triangulation is divided into two triangles (two-dimensional simplices) $\mathbb{I}_1 = [\mathbf{v}_1, \mathbf{v}_3, \mathbf{v}_4]$ and $\mathbb{I}_2 = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_4]$ (see Fig. 2.2).

Let's freely choose one of the simplex, say $\mathbb{I}_1$, and calculate the μ_2^q type lower bound:

$$\mu_2^q(\mathbb{I}_1) = \max_{\mathbf{v}_i \in \mathbb{V}(\mathbb{I}_1)} \left\{ f(\mathbf{v}_i) - L_p \max_{\mathbf{x} \in \mathbb{V}(\mathbb{I}_1)} \|\mathbf{x} - \mathbf{v}_i\|_q \right\}. \tag{2.12}$$

For $\mathbf{v}_1$ (or $\mathbf{v}_4$) the maximal distance

$$\max_{\mathbf{x} \in \mathbb{V}(\mathbb{I}_1)} \|\mathbf{x} - \mathbf{v}_1\|_q$$

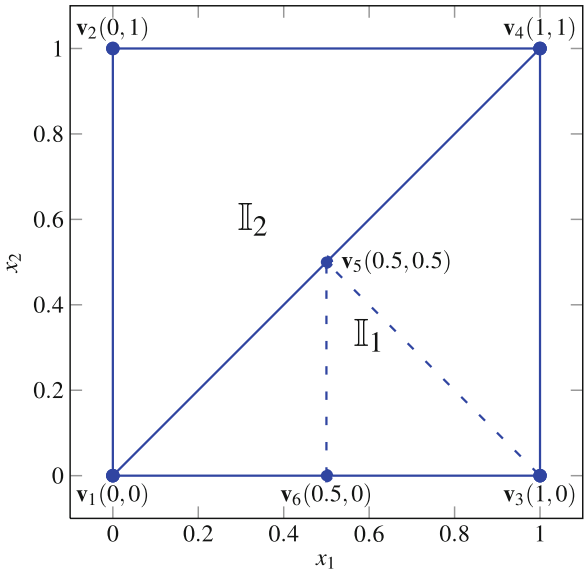

Fig. 2.2 Covering of square by simplices (*triangles*)

is between vertices $\mathbf{v}_1$ and $\mathbf{v}_4$, which depending on the used q-norm is equal to

$$\begin{aligned}
\|\mathbf{v}_1 - \mathbf{v}_4\|_1 &= (|0-1| + |0-1|) = 2,\\
\|\mathbf{v}_1 - \mathbf{v}_4\|_q &= (|0-1|^q + |0-1|^q)^{\frac{1}{q}} = 2^{\frac{1}{q}}, \quad 1 < q < \infty,\\
\|\mathbf{v}_1 - \mathbf{v}_4\|_\infty &= \max\{|0-1|, |0-1|\} = 1.
\end{aligned}$$

From (2.3), the product

$$L_p \max_{\mathbf{x}\in\mathbb{V}(\mathbb{I}_1)} \|\mathbf{x} - \mathbf{v}_i\|_q$$

depending on the used norm and the corresponding Lipschitz constant is equal to

$$\begin{aligned}
L_\infty\|\mathbf{v}_1 - \mathbf{v}_4\|_1 &= \max\left\{|f'_{x_1}(\mathbf{z})|, |f'_{x_2}(\mathbf{z})|\right\} \cdot 2,\\
L_p\|\mathbf{v}_1 - \mathbf{v}_4\|_q &= \left(|f'_{x_1}(\mathbf{z})|^p + |f'_{x_2}(\mathbf{z})|^p\right)^{\frac{1}{p}} \cdot 2^{\frac{1}{q}}, \quad \frac{1}{p} + \frac{1}{q} = 1,\ 1 < p, q < \infty,\\
L_1\|\mathbf{v}_1 - \mathbf{v}_4\|_\infty &= \left(|f'_{x_1}(\mathbf{z})| + |f'_{x_2}(\mathbf{z})|\right) \cdot 1,
\end{aligned}$$

where $\mathbf{z} = (z_1, z_2) \in \mathbb{D}$ is the point where the q-norm of the gradient of the objective function is largest. It can be noticed in Tables 2.1 and 2.2 that the points $\mathbf{z}$ may differ for various norms. However, for simplicity we assume that they are identical as is the case for most of the test problems in Tables 2.1 and 2.2.

Lemma 2.1 ([67]). *Let $p > 0$ and $a, b \geq 0$. Then*

$$(a+b)^p \leq \max\{1, 2^{p-1}\}(a^p + b^p).$$

Using Lemma 2.1 we get

$$\left(|f'_{x_1}(\mathbf{z})| + |f'_{x_2}(\mathbf{z})|\right) \cdot 1 \leq 2^{\frac{p-1}{p}} \left(|f'_{x_1}(\mathbf{z})|^p + |f'_{x_2}(\mathbf{z})|^p\right)^{\frac{1}{p}}. \tag{2.13}$$

Expressing $q = \frac{p}{p-1}$ and inserting into (2.13) we get

$$\begin{aligned}
\left(|f'_{x_1}(\mathbf{z})| + |f'_{x_2}(\mathbf{z})|\right) \cdot 1 &\leq \left(|f'_{x_1}(\mathbf{z})|^p + |f'_{x_2}(\mathbf{z})|^p\right)^{\frac{1}{p}} \cdot 2^{\frac{1}{q}}\\
&\leq \max\left\{|f'_{x_1}(\mathbf{z})|, |f'_{x_2}(\mathbf{z})|\right\} \cdot 2,
\end{aligned}$$

i.e.,

$$L_1\|\mathbf{v}_1 - \mathbf{v}_4\|_\infty \leq L_p\|\mathbf{v}_1 - \mathbf{v}_4\|_q \leq L_\infty\|\mathbf{v}_1 - \mathbf{v}_4\|_1. \tag{2.14}$$

From (2.14) it follows that for simplex $\mathbb{I}_1$ when the function values at vertices $\mathbf{v}_1$ and $\mathbf{v}_4$ are used, the best bound of μ_2^q type is with the ∞-norm and the corresponding

Lipschitz constant L_1, i.e.,

$$\mu_2^{\infty}(\mathbf{v}_i) = f(\mathbf{v}_i) - L_1\|\mathbf{v}_1 - \mathbf{v}_4\|_{\infty}, \tag{2.15}$$

where $i = 1$ or $i = 4$.

Calculating the bound over the simplex $\mathbb{I}_1$ vertex $\mathbf{v}_3$, it is easy to see that the distance to vertex $\mathbf{v}_1$ (or $\mathbf{v}_4$) independently on the used norm is equal to 1:

$$\begin{aligned}
\|\mathbf{v}_3 - \mathbf{v}_1\|_1 &= (|1-0| + |0-0|) = 1,\\
\|\mathbf{v}_3 - \mathbf{v}_1\|_q &= (|1-0|^q + |0-0|^q)^{\frac{1}{q}} = 1, \quad 1 < q < \infty,\\
\|\mathbf{v}_3 - \mathbf{v}_1\|_{\infty} &= \max\{|1-0|, |0-0|\} = 1.
\end{aligned}$$

Therefore when calculating the bound using the function value at vertex $\mathbf{v}_3$, the best bound is with the 1-norm and the corresponding L_{∞} Lipschitz constant:

$$\mu_2^{1}(\mathbf{v}_3) = f(\mathbf{v}_3) - L_{\infty}\|\mathbf{v}_3 - \mathbf{v}_1\|_1, \tag{2.16}$$

because $L_{\infty} \leq L_p$, $1 \leq p < \infty$.

In summary the best μ_2^q type bound over the simplex $\mathbb{I}_1$ is either with the 1-norm and the corresponding Lipschitz constant L_{∞} (2.16) or the ∞-norm and the corresponding Lipschitz constant L_1 (2.15).

It can be seen that for the $\mathbb{I}_2$ simplex and any simplex (isosceles right triangle $[\mathbf{v}_3, \mathbf{v}_4, \mathbf{v}_5]$, $[\mathbf{v}_3, \mathbf{v}_5, \mathbf{v}_6], \ldots$ (see Fig. 2.2)) later obtained using subdivision into two through the middle point of the longest edge, the best μ_2^q type bound are achieved, when either (2.15) or (2.16) is used.

Therefore to get tighter μ_2^q type bounds over two-dimensional simplices, it is useful to combine the 1-norm and the ∞-norm with the corresponding Lipschitz constants L_{∞} and L_1:

$$\mu_2^{1,\infty}(\mathbb{I}) = \max_{\mathbf{v}_i \in \mathbb{V}(\mathbb{I})} \left\{ f(\mathbf{v}_i) - \min\left\{ L_{\infty} \max_{\mathbf{x} \in \mathbb{V}(\mathbb{I})} \|\mathbf{x} - \mathbf{v}_i\|_1, L_1 \max_{\mathbf{x} \in \mathbb{V}(\mathbb{I})} \|\mathbf{x} - \mathbf{v}_i\|_{\infty} \right\} \right\}. \tag{2.17}$$

We observe that using μ_1^q type bound with the 1-norm and the ∞- norm also tighter bounds are achieved, comparing to the bound when just one of the norms is used:

$$\mu_1^{1,\infty}(\mathbb{I}) = \max_{\mathbf{v}_i \in \mathbb{V}(\mathbb{I})} f(\mathbf{v}_i) - \min\{L_{\infty}\delta_1(\mathbb{I}), L_1\delta_{\infty}(\mathbb{I})\}. \tag{2.18}$$

When the feasible region is a rectangle (not a square), after the initial vertex triangulation of the feasible region, the simplices (triangles) are not isosceles, therefore (2.14) is not always correct and proposed bound $\mu_2^{1,\infty}$ (2.17) is not

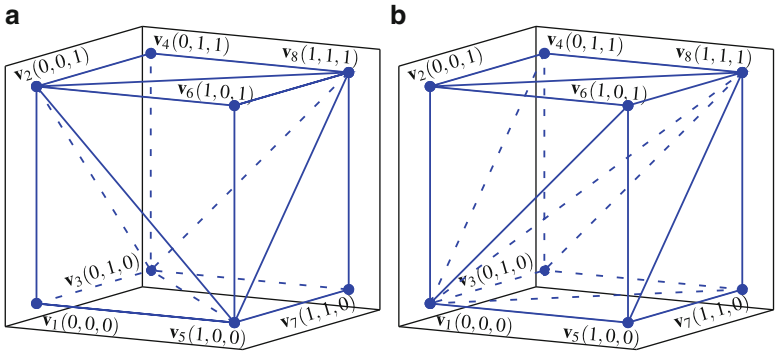

Fig. 2.3 Covering of a unit cube by simplices: (**a**) 5 tetrahedrons, (**b**) combinatorial triangulation with $3! = 6$ simplices

always better than one with a concrete norm. However, experimental investigation in Sect. 2.6 shows that the combined bound $\mu_2^{1,\infty}$ significantly improves optimization results for the rectangular feasible regions as well.

Let us extend our investigation when the feasible region is a (unit) cube $\mathbb{D} = [0,1]^3$. We investigate two different strategies of initial cube triangulation described in Sect. 1.3: face-to-face vertex triangulation by five simplices (see Fig. 2.3a) and combinatorial triangulation by six simplices (see Fig. 2.3b).

Let's start from the case, where the unit cube is triangulated into five tetrahedrons (see Fig. 2.3a). After such a face-to-face vertex triangulation we get four equal volume simplices and the middle one $[\mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_5, \mathbf{v}_8]$—with two times bigger volume. Let us investigate one of the four similar simplices, say the simplex $[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_5]$, and evaluate μ_2^q type bound using the function values at all vertices analogously as we did in the two-dimensional case.

When the bound is evaluated using the vertex $\mathbf{v}_1$, the maximal distance

$$\max_{\mathbf{x}\in[\mathbf{v}_2,\mathbf{v}_3,\mathbf{v}_5]} \|\mathbf{x}-\mathbf{v}_1\|_q$$

is equal to the distance to any of other vertices (lets take $\mathbf{v}_2$) and independently on the used norm is equal to 1:

$$\begin{aligned}
\|\mathbf{v}_1-\mathbf{v}_2\|_1 &= (|0-0|+|0-0|+|0-1|) = 1,\\
\|\mathbf{v}_1-\mathbf{v}_2\|_q &= (|0-0|^q+|0-0|^q+|0-1|^q)^{\frac{1}{q}} = 1, \quad 1<q<\infty,\\
\|\mathbf{v}_1-\mathbf{v}_2\|_\infty &= \max\{|0-0|,|0-0|,|0-1|\} = 1.
\end{aligned}$$

Therefore when using the function value at the vertex $\mathbf{v}_1$, the best bound is with the 1-norm and the corresponding Lipschitz constant L_∞:

$$\mu_2^1(\mathbf{v}_1) = f(\mathbf{v}_1) - L_\infty\|\mathbf{v}_1-\mathbf{v}_2\|_1. \tag{2.19}$$

When evaluating bounds using the remaining vertices $\mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_5$, the maximal distance, independently on the vertices (let's take $\mathbf{v}_2$)

$$\max_{\mathbf{x}\in[\mathbf{v}_1,\mathbf{v}_3,\mathbf{v}_5]} \|\mathbf{x}-\mathbf{v}_2\|_q$$

is between $\mathbf{v}_2$ and $\mathbf{v}_3$ (or between $\mathbf{v}_2$ and $\mathbf{v}_5$), which according to the used norm is equal to:

$$\begin{aligned}
\|\mathbf{v}_2-\mathbf{v}_3\|_1 &= (|0-0|+|0-1|+|1-0|)=2,\\
\|\mathbf{v}_2-\mathbf{v}_3\|_q &= (|0-0|^q+|0-1|^q+|1-0|^q)^{\frac{1}{q}}=2^{\frac{1}{q}}, \quad 1<q<\infty,\\
\|\mathbf{v}_2-\mathbf{v}_3\|_\infty &= \max\{|0-1|,|0-0|,|1-0|\}=1.
\end{aligned}$$

Therefore the following products of the Lipschitz constant and the corresponding norm may be

$$\begin{aligned}
L_\infty\|\mathbf{v}_2-\mathbf{v}_3\|_1 &= \max\left\{|f'_{x_1}(\mathbf{z})|,|f'_{x_2}(\mathbf{z})|,|f'_{x_3}(\mathbf{z})|\right\}\cdot 2,\\
L_p\|\mathbf{v}_2-\mathbf{v}_3\|_q &= \left(|f'_{x_1}(\mathbf{z})|^p+|f'_{x_2}(\mathbf{z})|^p+|f'_{x_3}(\mathbf{z})|^p\right)^{\frac{1}{p}}\cdot 2^{\frac{1}{q}}, 1<p,q<\infty,\\
L_1\|\mathbf{v}_2-\mathbf{v}_3\|_\infty &= \left(|f'_{x_1}(\mathbf{z})|+|f'_{x_2}(\mathbf{z})|+|f'_{x_3}(\mathbf{z})|\right)\cdot 1.
\end{aligned}$$

By analogy to (2.14), for the three-dimensional case, the inequality

$$\begin{aligned}
\left(|f'_{x_1}(\mathbf{z})|+|f'_{x_2}(\mathbf{z})|+|f'_{x_3}(\mathbf{z})|\right) &\le \left(|f'_{x_1}(\mathbf{z})|^p+|f'_{x_2}(\mathbf{z})|^p+|f'_{x_3}(\mathbf{z})|^p\right)^{\frac{1}{p}}\cdot 2^{\frac{1}{q}}\\
&\le \max\left\{|f'_{x_1}(\mathbf{z})|,|f'_{x_2}(\mathbf{z})|,|f'_{x_3}(\mathbf{z})|\right\}\cdot 2
\end{aligned} \tag{2.20}$$

is correct only when one of the partial derivatives $|f'_{x_i}(\mathbf{z})|$, $i=1,\ldots,3$ is equal or close to 0. Therefore for three-dimensional case $\mu_2^{1,\infty}$ (2.17) is not always better than μ_2^q with one concrete q-norm.

For the middle simplex $[\mathbf{v}_2,\mathbf{v}_3,\mathbf{v}_5,\mathbf{v}_8]$, the maximum distance from any vertex to other vertices is the same as in the previously investigated simplex $[\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3,\mathbf{v}_5]$: the distance between $\mathbf{v}_2$ and $\mathbf{v}_3$. It follows that $\mu_2^{1,\infty}$ is not always the best to this simplex too.

The investigated covering is not known in a general case for $n>3$. Therefore it is important to investigate the bounds using a general covering, e.g., combinatorial triangulation (see Fig. 2.3b).

We note that in such a covering, vertices $\mathbf{v}_1$ and $\mathbf{v}_8$ belong to all simplices and all simplices are of equal volume. Taking any of the simplices, say $[\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_4,\mathbf{v}_8]$ we can evaluate μ_2^q bounds. The maximal distance is between vertices $\mathbf{v}_1$ and $\mathbf{v}_8$, according to the used norm

$$\begin{aligned}
\|\mathbf{v}_1 - \mathbf{v}_8\|_1 &= (|0-1| + |0-1| + |0-1|) = 3,\\
\|\mathbf{v}_1 - \mathbf{v}_8\|_q &= (|0-1|^q + |0-1|^q + |0-1|^q)^{\frac{1}{q}} = 3^{\frac{1}{q}}, \quad 1 < q < \infty,\\
\|\mathbf{v}_1 - \mathbf{v}_8\|_\infty &= \max\{|0-1|, |0-1|, |0-1|\} = 1.
\end{aligned}$$

Therefore the following products are possibly depending on the used norms and the corresponding Lipschitz constants:

$$\begin{aligned}
L_\infty\|\mathbf{v}_1 - \mathbf{v}_8\|_1 &= \max\left\{|f'_{x_1}(\mathbf{z})|, |f'_{x_2}(\mathbf{z})|, |f'_{x_3}(\mathbf{z})|\right\} \cdot 3,\\
L_p\|\mathbf{v}_1 - \mathbf{v}_8\|_q &= \left(|f'_{x_1}(\mathbf{z})|^p + |f'_{x_2}(\mathbf{z})|^p + |f'_{x_3}(\mathbf{z})|^p\right)^{\frac{1}{p}} \cdot 3^{\frac{1}{q}}, \quad 1 < p, q < \infty,\\
L_1\|\mathbf{v}_1 - \mathbf{v}_8\|_\infty &= \left(|f'_{x_1}(\mathbf{z})| + |f'_{x_2}(\mathbf{z})| + |f'_{x_3}(\mathbf{z})|\right) \cdot 1.
\end{aligned}$$

In such a case, inequalities

$$\begin{aligned}
\left(|f'_{x_1}(\mathbf{z})| + |f'_{x_2}(\mathbf{z})| + |f'_{x_3}(\mathbf{z})|\right) &\le \left(|f'_{x_1}(\mathbf{z})|^p + |f'_{x_2}(\mathbf{z})|^p + |f'_{x_3}(\mathbf{z})|^p\right)^{\frac{1}{p}} \cdot 3^{\frac{1}{q}}\\
&\le \max\left\{|f'_{x_1}(\mathbf{z})|, |f'_{x_2}(\mathbf{z})|, |f'_{x_3}(\mathbf{z})|\right\} \cdot 3,
\end{aligned}$$

i.e.,

$$L_1\|\mathbf{v}_1 - \mathbf{v}_8\|_\infty \le L_p\|\mathbf{v}_1 - \mathbf{v}_8\|_q \le L_\infty\|\mathbf{v}_1 - \mathbf{v}_8\|_1 \tag{2.21}$$

hold. Therefore, the μ_2^q type bound calculated using vertex $\mathbf{v}_1$ (or $\mathbf{v}_8$) is best when the ∞-norm and the corresponding Lipschitz constant is used:

$$f(\mathbf{v}_1) - L_1\|\mathbf{v}_1 - \mathbf{v}_8\|_\infty. \tag{2.22}$$

Therefore after combinatorial triangulation, all simplices have two vertices, in which μ_2^q type bound with the infinity norm and the corresponding Lipschitz constant should be used.

Let's evaluate bound on the remaining vertices $\mathbf{v}_2$ and $\mathbf{v}_4$. The maximal distance from $\mathbf{v}_2$ is to vertex $\mathbf{v}_8$, and from vertex $\mathbf{v}_4$ to $\mathbf{v}_1$. In both cases the maximal distances are equal and depending on the used norm are

$$\begin{aligned}
\|\mathbf{v}_2 - \mathbf{v}_8\|_1 &= (|0-1| + |0-1| + |1-1|) = 2,\\
\|\mathbf{v}_2 - \mathbf{v}_8\|_q &= (|0-1|^q + |0-1|^q + |1-1|^q)^{\frac{1}{q}} = 2^{\frac{1}{q}}, \quad 1 < q < \infty,\\
\|\mathbf{v}_2 - \mathbf{v}_8\|_\infty &= \max\{|0-1|, |0-1|, |1-1|\} = 1.
\end{aligned}$$

The observed situation is analogous to (2.20), therefore by using this triangulation, $\mu_2^{1,\infty}$ bound over three-dimensional simplices is not always better than μ_2^q with one particular q-norm.

After the subdivision of simplices, the situation remains similar: not in all vertices combined bound $\mu_2^{1,\infty}$ is better than μ_2^q type bound with one concrete bound. Therefore for higher dimensional simplices, the real improvement is showed after experimental investigations (see Sect. 2.6).

2.4 Lipschitz Bound Based on Circumscribed Spheres

Calculation of tight Lipschitz bounds is usually computationally expensive. Therefore, it is important to investigate possibilities of bounds tighter than μ_1 (2.10) and μ_2 (2.11), but computationally less expensive than φ (2.8). This section is devoted to the Lipschitz bound over simplices based on circumscribed spheres [100]. The bound is often stronger than usually used trivial bounds and still computationally cheap, especially for low-dimensional problems for which calculation of the radius of the circumscribed sphere is cheap. To find the radius, $n+2$ determinants of $(n+1)\times(n+1)$-dimensionality matrices are calculated.

Proposition 2.2. *Let $f : \mathbb{D} \to \mathbb{R}, \mathbb{D} \subset \mathbb{R}^n$ be a Lipschitz continuous objective function, $\mathbb{I} \subset \mathbb{D}$ be an n-simplex with the set of vertices $\mathbb{V}(\mathbb{I})$, and $R_2(\mathbb{I})$ denotes the radius of the circumscribed n-sphere. Then*

$$\psi^2(\mathbb{I}) = \min_{\mathbf{v}\in\mathbb{V}(\mathbb{I})} f(\mathbf{v}) - L_2 R_2(\mathbb{I}), \tag{2.23}$$

underestimates $f(\mathbf{x}), \forall \mathbf{x} \in \mathbb{I}$.

Proof. An n-simplex $\mathbb{I}$ is covered by n-balls $\mathbb{O}_i$, $i = 1,\ldots,n+1$ such that the radius $R_2(\mathbb{I})$ is the same for all n-balls and the centers coincide with the vertices $\mathbf{v}_i$ of the simplex $\mathbb{I}$ (a two-dimensional example is shown in Fig. 2.4): $\forall \mathbf{x} \in \mathbb{I}\ \exists i$ such that $\mathbf{x} \in \mathbb{O}_i$. Then $\forall \mathbf{x} \in \mathbb{O}_i$

$$f(\mathbf{x}) \geq f(\mathbf{v}_i) - L_2 R_2(\mathbb{I}) \geq \min_{\mathbf{v}\in\mathbb{V}(\mathbb{I})} f(\mathbf{v}) - L_2 R_2(\mathbb{I}) = \psi^2(\mathbb{I}). \tag{2.24}$$

Let us derive a general formula to calculate the radius R_2 of the circumscribed n-sphere. A geometric construction for the radius of the circumscribed 2-sphere (circumscribed circle) is given by Pedoe [104]. The equation of the n-circumsphere of the n-simplex with the vertices $\mathbf{v}_1 = (v_{11},\ldots,v_{1n}),\ldots,\mathbf{v}_{n+1} = (v_{(n+1)1},\ldots,v_{(n+1)n})$ expressed in determinant form is

$$\begin{vmatrix} \sum_{i=1}^{n} x_i^2 & x_1 & x_2 & \ldots & x_n & 1 \\ \sum_{i=1}^{n} v_{1i}^2 & v_{11} & v_{12} & \ldots & v_{1n} & 1 \\ \sum_{i=1}^{n} v_{2i}^2 & v_{21} & v_{22} & \ldots & v_{2n} & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum_{i=1}^{n} v_{(n+1)i}^2 & v_{(n+1)1} & v_{(n+1)2} & \ldots & v_{(n+1)n} & 1 \end{vmatrix} = 0. \tag{2.25}$$

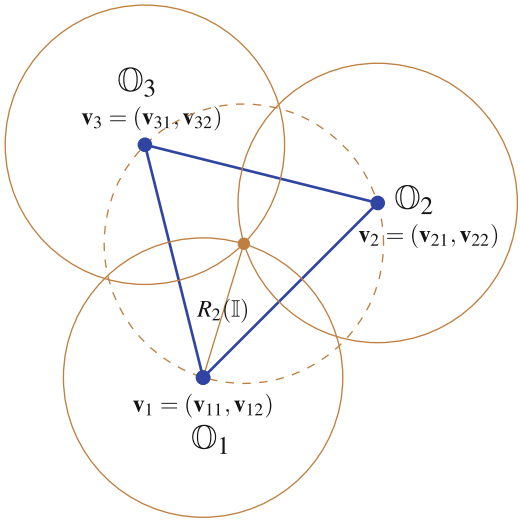

Fig. 2.4 Covering of a triangle (two-dimensional simplex) $\mathbb{I}$ by disks (2-balls) $\mathbb{O}_i, i = 1, \ldots, n+1$ of the same radius $R_2(\mathbb{I})$

Expanding the determinant by the first row we get

$$\mathbf{a}\left(\sum_{i=1}^{n} x_i^2\right) + \left(\sum_{i=1}^{n} \mathbf{b}_i x_i\right) + \mathbf{c} = 0, \tag{2.26}$$

where

$$\mathbf{a} = \begin{vmatrix} v_{11} & v_{12} & \ldots & v_{1n} & 1 \\ v_{21} & v_{22} & \ldots & v_{2n} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ v_{(n+1)1} & v_{(n+1)2} & \ldots & v_{(n+1)n} & 1 \end{vmatrix},$$

$$\mathbf{b}_1 = -\begin{vmatrix} \sum_{i=1}^{n} v_{1i}^2 & v_{12} & \ldots & v_{1n} & 1 \\ \sum_{i=1}^{n} v_{2i}^2 & v_{22} & \ldots & v_{2n} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum_{i=1}^{n} v_{(n+1)i}^2 & v_{(n+1)2} & \ldots & v_{(n+1)n} & 1 \end{vmatrix},$$

$\vdots$

$$\mathbf{b}_j = (-1)^j \begin{vmatrix} \sum_{i=1}^{n} v_{1i}^2 & v_{11} & \dots & v_{1(j-1)} & v_{1(j+1)} & \dots & v_{1n} & 1 \\ \sum_{i=1}^{n} v_{1i}^2 & v_{21} & \dots & v_{2(j-1)} & v_{2(j+1)} & \dots & v_{2n} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum_{i=1}^{n} v_{(n+1)i}^2 & v_{(n+1)1} & \dots & v_{(n+1)(j-1)} & v_{(n+1)(j+1)} & \dots & v_{(n+1)n} & 1 \end{vmatrix},$$

where $j = 2, \dots, n-1$,

$$\vdots$$

$$\mathbf{b}_n = (-1)^n \begin{vmatrix} \sum_{i=1}^{n} v_{1i}^2 & v_{11} & \dots & v_{1n-1} & 1 \\ \sum_{i=1}^{n} v_{1i}^2 & v_{21} & \dots & v_{2n-1} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum_{i=1}^{n} v_{(n+1)i}^2 & v_{(n+1)1} & \dots & v_{(n+1)n-1} & 1 \end{vmatrix},$$

$$\mathbf{c} = (-1)^{n+1} \begin{vmatrix} \sum_{i=1}^{n} v_{1i}^2 & v_{11} & \dots & v_{1n} \\ \sum_{i=1}^{n} v_{1i}^2 & v_{21} & \dots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} v_{(n+1)i}^2 & v_{(n+1)1} & \dots & v_{(n+1)n} \end{vmatrix}.$$

Completing the square for (2.26) gives:

$$\mathbf{a}\left(x_1 + \frac{\mathbf{b}_1}{2\mathbf{a}}\right)^2 + \dots + \mathbf{a}\left(x_n + \frac{\mathbf{b}_n}{2\mathbf{a}}\right)^2 - \frac{\mathbf{b}_1^2}{4\mathbf{a}} - \dots - \frac{\mathbf{b}_n^2}{4\mathbf{a}} + \mathbf{c} = 0$$

which is the n-circumsphere

$$\left(x_1 + \frac{\mathbf{b}_1}{2\mathbf{a}}\right)^2 + \dots + \left(x_n + \frac{\mathbf{b}_n}{2\mathbf{a}}\right)^2 = \frac{\mathbf{b}_1^2 + \dots + \mathbf{b}_n^2 - 4\mathbf{ac}}{4\mathbf{a}^2}$$

with the circumcenter

$$\mathbf{c} = (c_1, \ldots, c_n) = \left(-\frac{\mathbf{b}_1}{2\mathbf{a}}, \ldots, -\frac{\mathbf{b}_n}{2\mathbf{a}}\right)$$

and the circumradius

$$R_2 = \sqrt{\frac{\mathbf{b}_1^2 + \cdots + \mathbf{b}_n^2 - 4\mathbf{ac}}{4\mathbf{a}^2}}.$$

2.5 Tight Lipschitz Bound over Simplices with the 1-norm

In this section, φ^q type bound (2.8) based on the 1-norm (φ^1) is described. In this case, the lower bounding function is the envelope of n-dimensional pyramids and its maximum point is found by solving a system of linear equations [98]. In the case of the Euclidean norm, the lower bounding function is the envelope of n-dimensional cones and its maximum point can be found by solving a system of quadratic and linear equations. Therefore, the bound based on the first norm is less computationally expensive. Let us formulate two propositions, which are used for the evaluation of φ^1.

Proposition 2.3. *Let two n-pyramids $F_{\mathbf{v}_1}(\mathbf{x}) = f(\mathbf{v}_1) - L_\infty \|\mathbf{x} - \mathbf{v}_1\|_1$ and $F_{\mathbf{v}_2}(\mathbf{x}) = f(\mathbf{v}_2) - L_\infty \|\mathbf{x} - \mathbf{v}_2\|_1$ are defined and $f(\mathbf{v}_1) \leq f(\mathbf{v}_2)$. Then the intersection of pyramids is contained in a manifold of dimensionality $n-1$ defined by*

$$\sum_{i=1}^{n} d(\mathbf{v}_{1i}, \mathbf{v}_{2i}) - \frac{f(\mathbf{v}_2) - f(\mathbf{v}_1)}{L_\infty} = 0, \tag{2.27}$$

where

$$d(\mathbf{v}_{1i}, \mathbf{v}_{2i}) = \begin{cases} |2\mathbf{x}_i - \mathbf{v}_{1i} - \mathbf{v}_{2i}| & \text{when } \mathbf{v}_{1i} \leq \mathbf{x}_i \leq \mathbf{v}_{2i}, \\ 0 & \text{when } \mathbf{v}_{1i} = \mathbf{v}_{2i}, \\ |\mathbf{v}_{1i} - \mathbf{v}_{2i}| & \text{when } \mathbf{x}_i \notin [\mathbf{v}_{1i}, \mathbf{v}_{2i}], \end{cases}$$

and all points $\mathbf{x}$ in the intersection are closer to vertex $\mathbf{v}_1$ than to $\mathbf{v}_2$, i.e.,

$$||\mathbf{x} - \mathbf{v}_1||_1 \leq ||\mathbf{x} - \mathbf{v}_2||_1. \tag{2.28}$$

Proof. From equality $F_{\mathbf{v}_1}(\mathbf{x}) = F_{\mathbf{v}_2}(\mathbf{x})$ we get

$$\sum_{i=1}^{n} (-|\mathbf{x}_i - \mathbf{v}_{1i}| + |\mathbf{x}_i - \mathbf{v}_{2i}|) = \frac{f(\mathbf{v}_2) - f(\mathbf{v}_1)}{L_\infty}. \tag{2.29}$$

For each of the difference $|\mathbf{x}_i - \mathbf{v}_{2i}| - |\mathbf{x}_i - \mathbf{v}_{1i}|$, $i = 1, \ldots, n$ one case from the following possibilities is true:

$$\begin{cases} -2\mathbf{x}_i + \mathbf{v}_{1i} + \mathbf{v}_{2i} & \text{when } \mathbf{v}_{1i} \le \mathbf{x}_i \le \mathbf{v}_{2i}, \\ 2\mathbf{x}_i - \mathbf{v}_{1i} - \mathbf{v}_{2i} & \text{when } \mathbf{v}_{2i} \le \mathbf{x}_i \le \mathbf{v}_{1i}, \\ 0 & \text{when } \mathbf{v}_{1i} = \mathbf{v}_{2i}, \\ \mathbf{v}_{1i} - \mathbf{v}_{2i} & \text{when } \mathbf{x}_i > \mathbf{v}_{1i}, \mathbf{v}_{2i},\ \mathbf{v}_{1i} > \mathbf{v}_{2i}, \\ \mathbf{v}_{2i} - \mathbf{v}_{1i} & \text{when } \mathbf{x}_i > \mathbf{v}_{1i}, \mathbf{v}_{2i},\ \mathbf{v}_{1i} < \mathbf{v}_{2i}. \end{cases}$$

Therefore from (2.29) we get (2.27).

Because $f(\mathbf{v}_1) \le f(\mathbf{v}_2)$, from the equality $F_{\mathbf{v}_1}(\mathbf{x}) = F_{\mathbf{v}_2}(\mathbf{x})$ we get

$$-\|\mathbf{x} - \mathbf{v}_1\|_1 + \|\mathbf{x} - \mathbf{v}_2\|_1 = \frac{f(\mathbf{v}_2) - f(\mathbf{v}_1)}{L_\infty} \ge 0,$$

therefore (2.28) is true.

Proposition 2.4. *The Piyavskii type Lipschitz bound with the first norm φ^1 can be found solving a system of n linear equations.*

Proof. Let us define the numeration of vertices $\mathbf{v}_i$ so that $f(\mathbf{v}_1) \le f(\mathbf{v}_2) \le \cdots \le f(\mathbf{v}_{n+1})$. Intersection of pyramids $F_{\mathbf{v}_1}(\mathbf{x}) = f(\mathbf{v}_1) - L_\infty \|\mathbf{x} - \mathbf{v}_1\|_1$ and $F_{\mathbf{v}_i}(\mathbf{x}) = f(\mathbf{v}_i) - L_\infty \|\mathbf{x} - \mathbf{v}_i\|_1$, $i = 2, \ldots, n+1$ is $(n-1)$-manifold defined by (2.27). Taking into account (2.28), it is possible to consider only part of the manifold, which is defined by linear equation and constraints. Therefore it is possible to form a system of n linear equations defining intersections $F_{\mathbf{v}_1}(\mathbf{x}) = F_{\mathbf{v}_i}(\mathbf{x})$. If the solution of this system satisfies the constraints (see Example 2.1), then the lower bounding function attains it minimum at the solution point. If the solution of this system does not satisfy the constraints, then the minimum of the lower bounding function is the maximum of the function value at the intersections (see Example 2.2).

Example 2.1. Suppose the objective function is $f(\mathbf{x}) = -\sin(2x_1 + 1) - 2\sin(3x_2 + 2)$ and the feasible region $\mathbb{D} = [0, 1] \times [0, 1]$ is covered by two right-angled isosceles triangles $\mathbb{I}_1 = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$ and $\mathbb{I}_2 = [\mathbf{v}_1, \mathbf{v}_3, \mathbf{v}_4]$ (see Fig. 2.5).

Let us consider the first simplex $\mathbb{I}_1 = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$.

Because $f(\mathbf{v}_1) = f(0, 0) = -2.6601 < f(\mathbf{v}_2) = f(1, 0) = -1.9597 < f(\mathbf{v}_3) = f(1, 1) = 1.7767$, the intersection point is closer to vertex $\mathbf{v}_1$ and it is enough to find the intersection of pyramids $F_{\mathbf{v}_1} = F_{\mathbf{v}_2}$ and $F_{\mathbf{v}_1} = F_{\mathbf{v}_3}$.

The intersection of pyramids $F_{\mathbf{v}_1} = F_{\mathbf{v}_2}$ is:

$$-|\mathbf{x}_1 - 0| + |\mathbf{x}_1 - 1| - |\mathbf{x}_2 - 0| + |\mathbf{x}_2 - 0| = \frac{f(1, 0) - f(0, 0)}{6},$$

$$\text{however as } 0 \le \mathbf{x}_1 \le 1,$$

$$-\mathbf{x}_1 - \mathbf{x}_1 + 1 = \frac{f(1, 0) - f(0, 0)}{6},$$

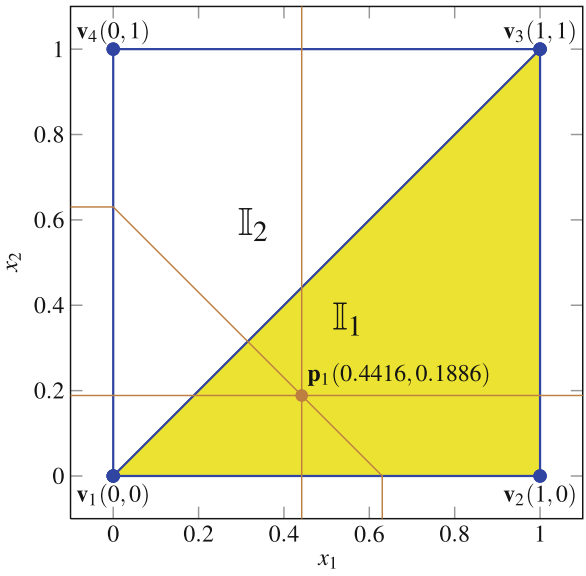

Fig. 2.5 Example 2.1: the projection of the intersection lines

$$\mathbf{x}_1 = 0.4416. \tag{2.30}$$

Analogously intersection of pyramids $F_{\mathbf{v}_1} = F_{\mathbf{v}_3}$ is:

$$-|\mathbf{x}_1 - 0| + |\mathbf{x}_1 - 1| - |\mathbf{x}_2 - 0| + |\mathbf{x}_2 - 1| = \frac{f(1,1) - f(0,0)}{6},$$

$$\begin{cases} \mathbf{x}_1 + \mathbf{x}_2 = -\frac{f(1,1)-f(0,0)}{12} + 1 = 0.6303, & \text{when } 0 \le \mathbf{x}_1 \le 1,\ 0 \le \mathbf{x}_2 \le 1, \\ \mathbf{x}_2 = 0.6303, & \text{when } \mathbf{x}_1 \le 0, \\ \mathbf{x}_1 = 0.6303, & \text{when } \mathbf{x}_2 \le 0. \end{cases} \tag{2.31}$$

From (2.28), (2.30), (2.31) $\Rightarrow$

$$\begin{cases} \mathbf{x}_1 = 0.4416 \\ \mathbf{x}_1 + \mathbf{x}_2 = 0.6303 \\ 0 \le \mathbf{x}_1 \le 0.5 \\ 0 \le \mathbf{x}_2 \le 1.0 \end{cases} \Rightarrow \text{intersection point } \mathbf{p}_1 = (0.4416, 0.1886).$$

Here we give four most significant digits. The lower bound $\varphi^1(\mathbb{I}_1)$ (see Fig. 2.6) is

$$\begin{aligned} \varphi^1(\mathbb{I}_1) &= F_{\mathbf{v}_i}(\mathbf{p}_1) = F_{\mathbf{v}_1}(\mathbf{p}_1) = f(\mathbf{v}_1) - L_\infty \|\mathbf{p}_1 - \mathbf{v}_1\|_1 \\ &= f(0,0) - 6(|0.4416 - 0| + |0.1886 - 0|) = -6.441. \end{aligned} \tag{2.32}$$

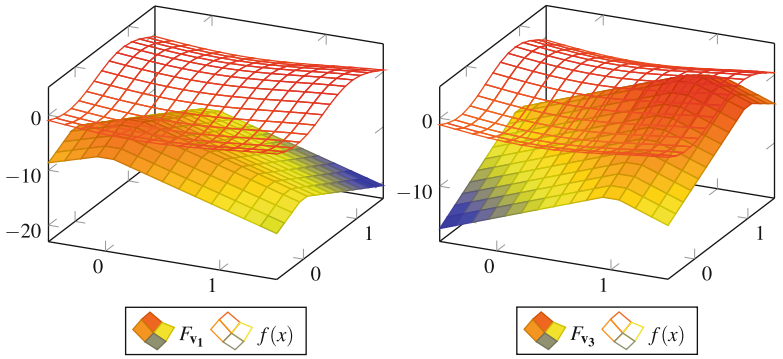

Fig. 2.6 Example 2.1: visualization of the lower bounding functions

Let us verify that this lower bound φ^1 is better than the simpler μ_2^q type bound with the first norm μ_2^1:

$$\begin{aligned}
\mu_2^1(\mathbf{v}_1) &= f(0,0) - 6 \max_{\mathbf{x}\in\mathbb{I}} \|\mathbf{x}-\mathbf{v}_1\|_1 = f(0,0) - 6\cdot 2 = -14.66,\\
\mu_2^1(\mathbf{v}_2) &= f(1,0) - 6 \max_{\mathbf{x}\in\mathbb{I}} \|\mathbf{x}-\mathbf{v}_2\|_1 = f(1,0) - 6\cdot 1 = -7.9597,\\
\mu_2^1(\mathbf{v}_3) &= f(1,1) - 6 \max_{\mathbf{x}\in\mathbb{I}} \|\mathbf{x}-\mathbf{v}_3\|_1 = f(1,1) - 6\cdot 2 = -10.223,\\
\mu_2^1(\mathbb{I}_1) &= \max\left\{\mu_2^1(\mathbf{v}_1), \mu_2^1(\mathbf{v}_2), \mu_2^1(\mathbf{v}_3)\right\} = -7.9597. \qquad (2.33)
\end{aligned}$$

From (2.32) and (2.33) $\Rightarrow \varphi^1(\mathbb{I}_1) > \mu_2^1(\mathbb{I}_1)$.

Example 2.2. The subdivision of simplices $\mathbb{I}_1$ and $\mathbb{I}_2$ produces four simplices $\mathbb{I}_3, \ldots, \mathbb{I}_6$ (see Fig. 2.7). Let us consider simplex $\mathbb{I}_3 = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_5]$. Since

$$f(\mathbf{v}_1) = -2.6601 < f(\mathbf{v}_2) = -1.9597 < f(\mathbf{v}_5) = -0.2077,$$

the intersection point is closer to vertex $\mathbf{v}_1$ and can be found from intersection of pyramids $F_{\mathbf{v}_1} = F_{\mathbf{v}_2}$ and $F_{\mathbf{v}_1} = F_{\mathbf{v}_5}$.

Intersection of pyramids $F_{\mathbf{v}_1} = F_{\mathbf{v}_2}$ is found in Example 2.1:

$$\mathbf{x}_1 = 0.4416. \qquad (2.34)$$

Intersection of pyramids $F_{\mathbf{v}_1} = F_{\mathbf{v}_5}$:

$$-|\mathbf{x}_1 - 0| + |\mathbf{x}_1 - 0.5| - |\mathbf{x}_2 - 0| + |\mathbf{x}_2 - 0.5| = \frac{f(0.5,0.5) - f(0,0)}{6},$$

$$\begin{cases}
\mathbf{x}_1 + \mathbf{x}_2 = -\dfrac{f(0.5,0.5) - f(0,0)}{12} + \frac{1}{2} = 0.2956, & \text{when } 0 \le \mathbf{x}_1, \mathbf{x}_2 \le 0.5\\
\mathbf{x}_2 = 0.2956, & \text{when } \mathbf{x}_1 \le 0,\\
\mathbf{x}_1 = 0.2956, & \text{when } \mathbf{x}_2 \le 0.
\end{cases} \qquad (2.35)$$

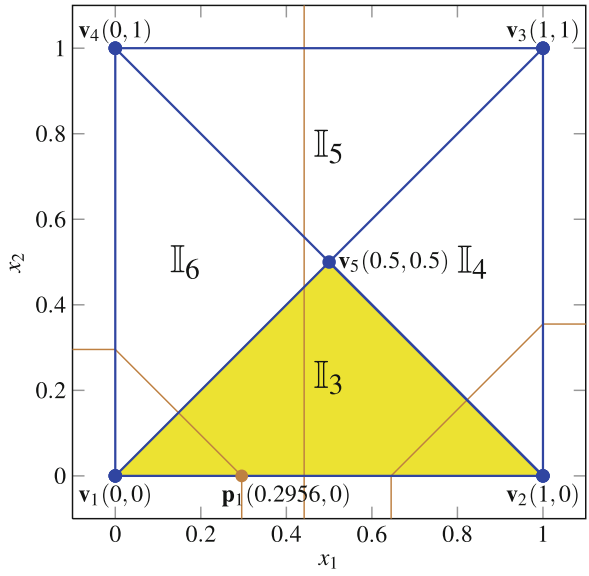

Fig. 2.7 Example 2.2: the projection of the intersection lines

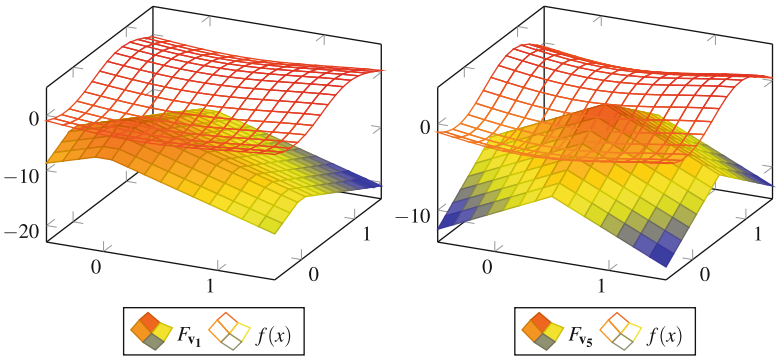

Fig. 2.8 Example 2.2: visualization of the lower bounding functions

From (2.28), (2.34), (2.35) $\Rightarrow$

$$\begin{cases} \mathbf{x}_1 = 0.4416 \\ \mathbf{x}_1 + \mathbf{x}_2 = 0.2956 \\ 0 \leq \mathbf{x}_1 \leq 0.5 \\ 0 \leq \mathbf{x}_2 \leq 0.5 \end{cases} \Rightarrow \varnothing \text{ (see Fig. 2.8)}$$

Therefore the lower bound is attained at the intersection line belonging to the lower bounding function (see Fig. 2.8), i.e.

$$\varphi^1(\mathbb{I}_3) = F_{\mathbf{v}_1}(\mathbf{p}_1),$$

where $\mathbf{p}_1$ is any point belonging to $\mathbf{x}_1 + \mathbf{x}_2 = 0.2956$ and $0 \le \mathbf{x}_1 \le 0.5$, $0 \le \mathbf{x}_2 \le 0.5$. Let $\mathbf{p}_1 = (0.2956, 0)$, then

$$\begin{aligned}\varphi^1(\mathbb{I}_3) &= F_{\mathbf{v}_1}(\mathbf{p}_1) = f(\mathbf{v}_1) - L_\infty \|\mathbf{p}_1 - \mathbf{v}_1\|_1 \\ &= f(0,0) - 6(|0.2956 - 0| + |0 - 0|) = -4.4339. \end{aligned} \tag{2.36}$$

Let us verify that this φ^q type bound with the first norm φ^1 is better than μ_2^1:

$$\begin{aligned}\mu_2^1(\mathbf{v}_1) &= f(0,0) - 6 \max_{\mathbf{x}\in\mathbb{I}} \|\mathbf{x} - \mathbf{v}_1\|_1 = f(0,0) - 6 \cdot 1 = -8.6601, \\ \mu_2^1(\mathbf{v}_2) &= f(1,0) - 6 \max_{\mathbf{x}\in\mathbb{I}} \|\mathbf{x} - \mathbf{v}_2\|_1 = f(1,0) - 6 \cdot 1 = -7.9597, \\ \mu_2^1(\mathbf{v}_5) &= f(0.5,0.5) - 6 \max_{\mathbf{x}\in\mathbb{I}} \|\mathbf{x} - \mathbf{v}_5\|_1 = f(0.5,0.5) - 6 \cdot 1 = -6.2077, \\ \mu_2^1(\mathbb{I}_3) &= \max\left\{\mu_2^1(\mathbf{v}_1), \mu_2^1(\mathbf{v}_2), \mu_2^1(\mathbf{v}_5)\right\} = -6.2077. \end{aligned} \tag{2.37}$$

From (2.36) and (2.37) $\Rightarrow \varphi^1(\mathbb{I}_3) > \mu_2^1(\mathbb{I}_3)$.

2.6 Branch-and-Bound with Simplicial Partitions and Various Lipschitz Bounds

Branch-and-bound algorithms differ by selection strategy of the node to process, branching and bound calculation. A general branch-and-bound algorithm was presented in Algorithm 1. In this section we use simplicial partitioning, combinatorial covering of the rectangular feasible region, best-first selection, subdivision of simplices through the middle of the longest edge, the minimum of the function values at the vertices of simplices as the upper bound for the minimum and various lower bounds μ_2^q, φ^1, ψ^2, as well as aggregates of them with different norms and the corresponding Lipschitz constants. Simplicial branch-and-bound algorithm is shown in Algorithm 4.

Algorithm 4 Simplicial branch-and-bound algorithm with various Lipschitz bounds

1: Cover $\mathbb{D}$ by $\mathbb{L} \leftarrow \left\{\mathbb{L}_j | \mathbb{D} = \bigcup_{j=1}^{n!} \mathbb{L}_j\right\}$ using combinatorial triangulation
2: $\mathbb{S} \leftarrow \varnothing, UB(\mathbb{D}) \leftarrow \infty$
3: **while** ($\mathbb{L} \neq \varnothing$) **do**
4: Choose $\mathbb{I} \in \mathbb{L}$ using selection strategy, $\mathbb{L} \leftarrow \mathbb{L} \backslash \{\mathbb{I}\}$
5: $UB(\mathbb{D}) \leftarrow \min\{UB(\mathbb{D}), \min_{\mathbf{v}\in\mathbb{V}(\mathbb{I})} f(\mathbf{v})\}$
6: $\mathbb{S} \leftarrow \arg\min\left\{f(\mathbb{S}), \min_{\mathbf{v}\in\mathbb{V}(\mathbb{I})} f(\mathbf{v})\right\}$
7: Calculate $LB(\mathbb{I})$ using different Lipschitz bounds
8: **if** ($LB(\mathbb{I}) < UB(\mathbb{D}) - \varepsilon$) **then**
9: Branch $\mathbb{I}$ into 2 simplices: $\mathbb{I}_1, \mathbb{I}_2$
10: $\mathbb{I} \leftarrow \mathbb{I} \cup \{\mathbb{I}_1, \mathbb{I}_2\}$
11: **end if**
12: **end while**

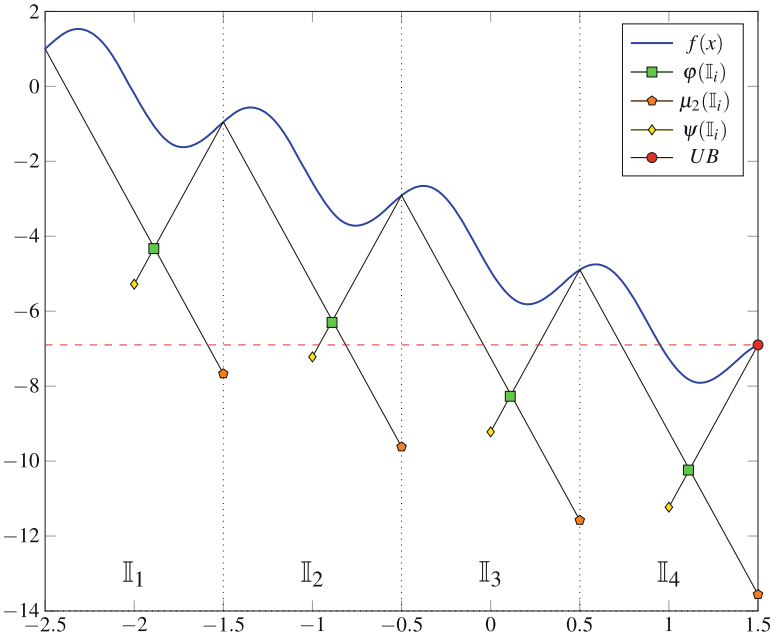

Fig. 2.9 Second step of branch-and-bound algorithm with simplicial partitions ($n = 1$)

The branch-and-bound process is illustrated using an one-dimensional example in Fig. 2.9.

Example 2.3. Let $\mathbb{D} = [-2.5, 1.5]$, $f(x)$ is Lipschitz function (2.9). As the example is one dimensional, initial covering is trivial: the feasible region is the initial simplex. Then it is divided into two through the middle point (the first step) and both are subdivided into two (the second step) again. At this step of the search there are four simplices $\mathbb{I}_i, i = 1, \ldots, 4$ (see Fig. 2.9). Three different lower bounds of different type: φ (—■—), μ_2 (—●—) and ψ (—◆—) are shown. We remind that in one-dimensional case all norm are equal; therefore there is no need to specify what norms are used.

The bounds μ_2 over all simplices $\mathbb{I}_i : \mu_2(\mathbb{I}_i) < UB, i = 1, \ldots, 4$, however using ψ type lower bound we get $\psi(\mathbb{I}_1) > UB$ and using the φ type lower bound we get $\varphi(\mathbb{I}_1) > UB$ and $\varphi(\mathbb{I}_2) > UB$. Therefore, two simplices $(\mathbb{I}_1, \mathbb{I}_2)$ in the case of φ type bound and only one simplex $\mathbb{I}_1$ in the case of ψ bound can be discarded from the further search. Only non-discarded simplices will be subdivided further. Therefore we can detect subregions that cannot contain the global optimizer using φ and ψ type bounds faster.

Let us perform and discuss computational experiments with simplicial branch-and-bound algorithm using different Lipschitz bounds. Various test problems ($n \geq 2$) for global optimization from [49, 60, 82] have been used in our experiments. The

full description of the used test problems is given in Appendix A. For ($n = 2, 3$) we use the same precision as used by Hansen and Jaumard [49]. The speed of global optimization has been estimated using the number of function evaluations criterion. Average values for each dimensionality are shown in bold font in following tables.

For two-dimensional test problems no single norm and the corresponding Lipschitz constant are the best for all test functions. However theoretical (see Sect. 2.3) and experimental investigation (see Table 2.3) shows that using the Lipschitz bound based on two extreme (infinite and first) norms $\mu_2^{1,\infty}$ (2.17) gives the best results. On average the number of function evaluations is by 21% smaller than when the bound based on the traditional Euclidean norm μ_2^2 is used.

It was shown in Sect. 2.3 that when the feasible region is a hyper-rectangle and $n \geq 3$, the bound $\mu_2^{1,\infty}$ is not always best. For some optimization problems better results are obtained using other norms. The results of experimental investigation suggest us include in μ_2^2 bound in the aggregate bound:

$$\mu_2^{1,2,\infty}(\mathbb{I}) = \max_{\mathbf{v} \in \mathbb{V}(\mathbb{I})} \{f(\mathbf{v}) - K(\mathbb{V}(\mathbb{I}))\}, \tag{2.38}$$

where

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

For $n = 2$ test problems the Euclidean norm is useful only for rectangular feasible regions. The feasible region of only one (No. 10) of the thirteen test problems is rectangle. Only for this test problem $\mu_2^{1,2,\infty}$ improves the result comparing with the result when $\mu_2^{1,\infty}$ was used.

For test problems of higher dimensionality ($n \geq 3$) the bound $\mu_2^{1,2,\infty}$ improves the results almost for all test problems comparing with $\mu_2^{1,\infty}$ bound. On average the number of function evaluations is about 5% smaller than with $\mu_2^{1,\infty}$ bound and about 52% than in the case the Euclidean norm is used alone.

We observe that using only one single norm μ_2^q for some test problems the solution has not been found after a huge number of function evaluations—10, 000, 000.

It was mentioned that the Piyavskii type bound (φ) is the tightest (most accurate) Lipschitz bound over some subregion, when the function values at some feasible points and the Lipschitz constant L is known. However calculation of such a bound is a hard optimization problem involving solving a system of quadratic [112] or quadratic and linear [89] equations. Computationally less expensive Piyavskii type bound based on the first norm φ^1 was described in Sect. 2.5.

On the first part of the experiments the bound φ^1 was compared with μ_2^1. From the results presented in Table 2.4, we see that for all test problems better results are achieved using φ^1 than μ_2^1. Depending on the dimensionality of test problems, the number of function evaluations is from 4 to 30% smaller than with a simpler bound.

Table 2.3 The number of function evaluations using μ_2^1, μ_2^2, μ_2^∞, $\mu_2^{1,\infty}$, and $\mu_2^{1,2,\infty}$ bounds

#	μ_2^1	μ_2^2	μ_2^∞	$\mu_2^{1,\infty}$	$\mu_2^{1,2,\infty}$
1	2019	1356	970	970	970
2	321	299	246	216	216
3	10700	8935	10825	7539	7539
4	40	52	72	8	8
5	61	126	153	61	61
6	2864	2046	1953	1751	1751
7	34814	23100	24098	20200	20200
8	528	729	1042	521	521
9	73333	42844	46233	40314	40314
10	2421	3055	4397	2369	2297
11	6783	6986	9426	5654	5654
12	29369	37756	58690	29357	29357
13	44908	37774	22262	22241	22241
mean	**16012**	**12697**	**13874**	**10092**	**10087**
14	4157824	6043338	5693624	4157824	4157740
15	51730	14140	3864	3864	3864
16	>10000000	>10000000	3145728	3145728	3145728
17	1663261	2702422	3721012	1596633	1596633
18	47080	19632	38888	38792	18720
19	25292	50812	60088	24748	24748
20	904186	713398	777482	777478	668510
mean	**>2407053**	**>2791963**	**1920098**	**1392152**	**1373706**
21	>10000000	>10000000	6496801	6496791	6264147
22	256557	225298	721331	239217	175107
23	556488	1048292	5347547	556488	556412
24	1031808	1066128	5353683	1031808	1031808
25	573359	1066128	5359481	573281	573025
26	2036200	1565127	5801659	1852688	1253836
27	572414	496904	3818039	549730	438141
28	408540	489831	1933931	384746	331977
mean	**>1929421**	**>1994714**	**4354059**	**1460594**	**1328057**
29	>10000000	7914387	1917124	1917124	1917092
30	6936368	8284881	>10000000	6932546	6239743
31	6490797	8535473	>10000000	6482327	6099520
mean	**>7809055**	**8244914**	**>7305708**	**5110666**	**4752118**
32	>10000000	6269636	823320	823320	821892
33	1926497	7419819	>10000000	1926497	1919569
mean	**>5963249**	**6844728**	**>5411660**	**1374909**	**1370731**

Previous investigations have shown (see Table 2.3) that there are no single norm and the corresponding Lipschitz constant best for all test problems. Therefore, Piyavskii type bound with the first norm φ^1 is not optimal for all test problems. From the results shown in Table 2.4, it can be seen that φ^1 bound gives worse results than $\mu_2^{1,2,\infty}$ for the test problems, for which the bounds with the first norm

Table 2.4 The number of function evaluations using μ_2^1, φ^1, $\mu_2^{1,2,\infty}$, and $\varphi^1\mu_2^{2,\infty}$ bounds

#	μ_2^1	φ^1	$\mu_2^{1,2,\infty}$	$\varphi^1\mu_2^{2,\infty}$
1	2019	1668	970	967
2	321	215	216	188
3	10700	8264	7539	6653
4	40	28	8	8
5	61	39	61	39
6	2864	2465	1751	1723
7	34814	26792	20200	19171
8	528	381	521	380
9	73333	54630	40314	38860
10	2421	1866	2297	1807
11	6783	5416	5654	4789
12	29369	21160	29357	21153
13	44908	33917	22241	22094
mean	**16012**	**12065**	**10087**	**9064**
14	4157824	3423524	4157740	3423480
15	51730	41212	3864	3863
16	>10000000	>10000000	3145728	3145728
17	1663261	1410165	1596633	1363004
18	47080	34464	18720	16884
19	25292	20876	24748	20487
20	904186	733234	668510	547622
mean	**>2407053**	**>2237639**	**1373706**	**1217295**
21	>10000000	>10000000	6264147	6262233
22	256557	240901	175107	167217
23	556488	536356	556412	536280
24	1031808	988493	1031808	988493
25	573359	544034	573025	543712
26	2036200	1867961	1253836	1176018
27	572414	546240	438141	420417
28	408540	383084	331977	314023
mean	**>1929421**	**>1888384**	**1328057**	**1301049**
29	>10000000	>10000000	1917092	1916941
30	6936368	6730455	6239743	6064924
31	6490797	5641951	6099520	5940304
mean	**>7809055**	**>7682866**	**4752118**	**4640723**
32	>10000000	>10000000	821892	821892
33	1926497	1875911	1919569	1868983
mean	**>5963249**	**>5937956**	**1370731**	**1345438**

are not efficient. Therefore, in order to reduce the number of function evaluations, it is appropriate to replace μ_2^1 bound with more tight bound with the same norm (φ^1) in an aggregate. In such a case the aggregate bound may be composed of $\mu_2^{2,\infty}$ and φ^1 [99]:

$$\varphi^1\mu_2^{2,\infty}(\mathbb{I}) = \max\left\{\varphi^1(\mathbb{I}), \mu_2^{2,\infty}(\mathbb{I})\right\} \tag{2.39}$$

$$= \max\left\{\min_{\mathbf{x}\in\mathbb{I}}\left(\max_{\mathbf{v}\in\mathbb{V}(\mathbb{I})}\left\{f(\mathbf{v})-L_\infty\|\mathbf{x}-\mathbf{v}\|_1\right\}\right), \max_{\mathbf{v}\in\mathbb{V}(\mathbb{I})}\left\{f(\mathbf{v})-K'(\mathbb{V}(\mathbb{I}))\right\}\right\},$$

where

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

On average the number of function evaluations with this aggregate bound are 9% smaller compared to that of $\mu_2^{1,2,\infty}$, and by 25% smaller compared to the results with φ^1.

The experimental results of simplicial branch-and-bound algorithm with the bound ψ^2 (2.23) and with the bound μ_2^2 (2.10) based on the same Euclidean norm are shown in Table 2.5. The ratio $r_{\psi^2<\mu_2^2}$ shows how often the bound ψ^2 is better (more tight) than the bound μ_2^2. For most of the test problems the number of function evaluations is smaller when the bound ψ^2 is used and on the average it is by 47% smaller for ($n = 2, 3$) test problems and by 26% smaller for ($n \geq 4$) test problems than when the bound μ_2^2 is used. However, the bound ψ^2 is not always better comparing with μ_2^2 (the ratio $r_{\psi^2<\mu_2^2}$ is not always equal to 1), therefore for computationally expensive functions it might be worthwhile include in the bound ψ^2 in the aggregate bound $\varphi^1\mu_2^{2\infty}$ (2.39) resulting in improved aggregate bound $\varphi^1\psi^2\mu_2^{2,\infty}$ [100]:

$$\varphi^1\psi^2\mu_2^{2,\infty}(\mathbb{I}) = \max\left\{\varphi^1\mu_2^{2,\infty}(\mathbb{I}), \psi^2(\mathbb{I})\right\}. \tag{2.40}$$

The number of function evaluations are on average 20% smaller when the improved aggregate bound $\varphi^1\psi^2\mu_2^{2,\infty}$ is used than when the aggregate bound $\varphi^1\mu_2^{2,\infty}$ is used (Table 2.5).

The value of the objective function at the same point is required when computing bounds over neighbor simplices. If this value is evaluated for the parent simplex, it is not necessary to evaluate it again. However, in multidimensional case the same points may be encountered when subdividing different simplices. One possibility is to maintain a list of already evaluated points and before evaluating a point to check it has already been evaluated. Vertex verification reduces the number of function evaluations essentially, and also increases computational time; therefore it is more suitable in the case of computationally expensive functions. The number of function evaluations when improved aggregate bound is used and vertices are verified $\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$ are shown in Table 2.5 too.

Let us compare the simplicial branch-and-bound algorithm with aggregate bound (2.40) and vertex verifications $\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$ to other well-known algorithms

Table 2.5 The number of function evaluations using μ_2^2 and ψ^2 type bounds, aggregate $\varphi^1\mu_2^{2,\infty}$ and $\varphi^1\psi^2\mu_2^{2,\infty}$ bounds and $\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$

#	μ_2^2	ψ^2	$\varphi^1\mu_2^{2,\infty}$	$\varphi^1\psi^2\mu_2^{2,\infty}$	$\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$	r_{ψ^2/μ_2^2}
1	1356	856	967	716	412	0.93
2	299	286	188	185	122	0.39
3	8935	5048	6653	4843	2551	1.00
4	52	128	8	8	5	0.02
5	126	155	39	39	36	0.06
6	2046	1284	1723	1241	691	0.99
7	23100	12547	19171	12195	6240	0.99
8	729	479	380	341	212	0.96
9	42844	22038	38860	21724	11049	1.00
10	3055	1734	1807	1495	830	0.98
11	6986	3915	4789	3598	1931	0.00
12	37756	19211	21153	16938	8926	1.00
13	37774	20366	22094	18737	9855	1.00
mean	**12697**	**6773**	**9064**	**6312**	**3297**	**0.72**
14	6043338	2642392	3423480	2355490	495711	0.99
15	14140	11928	3863	3784	1030	0.64
16	>10000000	6325269	3145728	3145728	536761	1.00
17	2702422	1131286	1363004	1020782	229049	0.95
18	19632	14368	16884	12032	3091	0.75
19	50812	20776	20487	17105	4684	0.91
20	713398	281998	547622	262311	55605	0.98
mean	**>2791963**	**1489717**	**1217295**	**973890**	**189419**	**0.89**
21	>10000000	7359930	6262233	5349848	540697	0.64
22	225298	182447	167217	120144	13707	0.59
23	1048292	635716	536280	450084	44421	0.67
24	1066128	635832	988493	563835	68502	0.67
25	1066128	635922	543712	465955	44991	0.67
26	1565127	965474	1176018	749518	52078	0.65
27	496904	426493	420417	333568	5769	0.64
28	489831	918077	314023	271614	37981	0.24
mean	**>1994714**	**1469986**	**1301049**	**1038071**	**101018**	**0.60**
29	7914387	5826460	1916941	1633849	84406	0.57
30	8284881	8079412	6064924	4590448	162989	0.52
31	8535473	11291062	5940304	5192437	256963	0.47
mean	**8244914**	**8398978**	**4640723**	**3805578**	**168119**	**0.52**
32	6269636	1623674	821892	524940	9840	0.57
33	7419819	6818423	1868983	1685793	25398	0.50
mean	**6844728**	**4221049**	**1345438**	**1105367**	**17619**	**0.53**

for Lipschitz optimization, described by Hansen and Jaumard [49]. Two classes of algorithms are considered:

1. Algorithms using a single lower-bounding function, i.e., variants of Piyavskii algorithm [111, 112]:
   - Mladineo (Mla) [89],
   - Jaumard, Herrmann, and Ribault (JHR) [61],
   - Wood (Wood) [141, 142].
2. Branch-and-bound algorithms:
   - Simplicial branch-and-bound algorithm with aggregate Lipschitz bound $\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$,
   - Galperin (Gal85, Gal88) [37, 38],
   - Pinter (Pint) [107],
   - Meewella and Mayne (MM) [86],
   - Gourdin, Hansen, and Joumard (GHJ) [45].

The comparison of the algorithms is based on the number of function evaluation criterion. The number of function evaluations are presented in Table 2.6.

It is mentioned in [49] that the results for all algorithms may be obtained only when the required precision (ε) is not too restrictive. Even so, some problems cannot be solved by some of the algorithms in reasonable computational time and/or memory size. In the experiments we apply the precision used in [49]. The number of function evaluations are smallest, when the algorithms of Mladineo and of Jaumard, Herrmann, and Ribault are used. However, these algorithms belong to the first class of algorithms and require a longer computational time. The branch-and-bound algorithms require a larger number of function evaluations, but much shorter computational time. The best numbers for branch-and-bound algorithms are shown in bold. The performance of the simplicial branch-and-bound algorithm with aggregate bound $\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$ is similar to that of the best branch-and-bound algorithm (GHJ) and often it is even better.

2.7 Parallel Branch-and-Bound with Simplicial Partitions

Global optimization algorithms are computationally intensive and therefore parallel computing is important [16, 19, 28, 88, 94]. For some parallel Lipschitz global optimization methods, conditions, which guarantee considerable speedup with respect to the sequential version of the algorithm, are established [122, 133, 134].

In this section two parallel algorithms based on the branch-and-bound technique are described [103]. OpenMP [13, 14] and MPI [47, 131] were used to implement parallel versions of the algorithm. The parallel algorithms belong to the second type of parallelism according to [40]. The parallel OpenMP versions are implemented as

Table 2.6 Comparison with well-known Lipschitz optimization algorithms

	Iterative algorithms			Branch-and-bound algorithms					
#	Mla	JHR	Wood	$\widehat{\varphi^1\psi^2\mu_2^{2,\infty}}$	Gal85	Gal88	Pint	MM	GHJ
1	320	323	5528	**412**	3553	1713	3807	1749	643
2	80	80	2861	**122**	1036	577	1762	744	167
3	2066	2066	70955	**2551**	24214	16089	28417	10839	3531
4	6	6	157	**5**	106	73	1527	94	45
5	41	41	209	**36**	430	217	907	424	73
6	548	548	14740	**691**	7729	2929	7772	2684	969
7	–	5088	183759	**6240**	43123	34705	62917	22799	7969
8	177	177	1403	**212**	2113	1289	2272	964	301
9	–	8838	309763	**11049**	57814	49873	88932	53549	13953
10	673	673	18613	**830**	8508	5628	9022	3814	1123
11	1613	1613	53348	**1931**	18235	12737	20312	9224	2677
12	–	8414	470200	**8926**	63088	56177	105572	45389	12643
13	–	9617	–	**9855**	65536	59049	109227	35949	15695
14	>460	>41700	–	495711	5383113	3886897	–	–	**215061**
15	>290	9363	–	**1030**	635909	347075	–	–	24249
16	>290	>12000	–	**536761**	15620627	–	–	–	1297205
17	>280	>14400	–	**229049**	12481708	–	–	–	268279
18	>690	1309	–	**3091**	46411	23765	–	–	3219
19	446	445	–	**4684**	35463	18669	–	–	7177

synchronous single pool (SSP) (one list of candidate simplices is maintained) and the MPI versions as asynchronous multiple pool (AMP) (different processors use separate lists). The synchronous algorithms are executed in phases.

A sequential branch-and-bound algorithm with simplicial partitions and various Lipschitz bounds was described in Sect. 2.6.

Two OpenMP versions were implemented: with a vertex point verification in the global vertex set of all previously evaluated function values at the midpoints of the longest edge and without it. If the function value at the new vertex point has not been evaluated before, the function is evaluated. In the other case the previously evaluated function value is used to calculate bounds. Therefore it is possible to reduce the number of function evaluations avoiding several evaluations at the same point.

The OpenMP version of the parallel branch-and-bound algorithm with vertex point verification is shown in Algorithm 5. Data parallelism is used. The feasible region $\mathbb{D}$ is subsequently divided into a set of simplices $\mathbb{L} = \{\mathbb{L}_j\}$. In C/C++, OpenMP directives are specified by using the #pragma mechanism. The #pragma directives offer a way for each compiler to offer machine- and operating system-specific features while retaining overall compatibility with the C and C++ languages. Directive "for" specifies that the iterations of the loop immediately following it must be executed in parallel by different threads. "schedule(static)" describes that iterations of the loop are evenly (if possible) divided and then

Algorithm 5 OpenMP version of parallel branch-and-bound algorithm

```
1: Cover 𝔻 by 𝕃 = {𝕃_j | 𝔻 ⊆ ⋃_{j=1}^{n!} 𝕃_j } using face-to-face vertex triangulation.
2: 𝕊 ← ∅, UB(𝔻) ← ∞, 𝕍(𝔻) = {v_j : j = 1, ..., 2^n}
3: while (𝕃 ≠ ∅) do
4:    #pragma omp parallel private (LB)
5:    #pragma omp for schedule(static)
6:    for (j = 1; j <= |𝕃|; j++) do
7:       𝕀 ← 𝕃, 𝕃 ← ∅
8:       #pragma omp critical (UB(𝔻))
9:       UB(𝔻) ← min {UB(𝔻), min_{v∈𝕍(𝕀_j)} {f(v)}}
10:      #pragma omp critical (𝕊)
11:      𝕊 ← arg min {f(𝕊), min_{v∈𝕍(𝕀_j)} f(v)}
12:      Calculate LB(𝕀_j) using different Lipschitz bounds
13:      if (LB(𝕀_j) < UB(𝔻) − ε) then
14:         Branch 𝕀_j into 2 simplices: 𝕀_1, 𝕀_2
15:         if v_j ∉ 𝕍(𝔻) then
16:            𝕍(𝔻) ← 𝕍(𝔻) ∪ {v_j}
17:         end if
18:         #pragma omp critical (𝕃)
19:         𝕃 ← 𝕃 ∪ {𝕀_1, 𝕀_2}
20:      end if
21:   end for
22: end while
```

statically assigned to threads. The directive "critical" specifies a region of code that must be executed by only one thread at a time.

Each simplex is evaluated by checking if it can contain an optimal solution. For this purpose, the lower bound $LB(\mathbb{I}_j)$ for the objective function f is evaluated over each simplex and compared with the upper bound $UB(\mathbb{D})$ for the minimum value over the feasible region. If $UB(\mathbb{D}) - LB(\mathbb{I}_j) > \varepsilon$, then the simplex $\mathbb{I}_j$ cannot contain a function value better (smaller) than one already found by more than the given precision ε, and therefore it is rejected. Otherwise it is inserted into the set of unexplored simplices $\mathbb{L}$. The algorithm terminates when there are no more potential simplices to investigate.

Two MPI versions with static load balancing were implemented using a parallel branch-and-bound template [2]: with interchange of the best currently found values among processors and without. When the template is used, only algorithm-specific rules should be described by the user and the standard parts are implemented in the template. The MPI library is used for the underlying communications.

Static load balancing is used: tasks are initially distributed evenly (if possible) among p parallel processes and then the processes work independently and do not exchange any later generated tasks. Each parallel process runs the same algorithm, which is shown in Algorithm 6. The algorithm is very similar to the sequential Algorithm 4. The differences are:

Algorithm 6 MPI version of parallel branch-and-bound algorithm

1: Cover $\mathbb{D}$ by $\mathbb{L} = \{\mathbb{L}_j | \mathbb{D} \subseteq \bigcup_{j=1}^{n!} \mathbb{L}_j\}$ using face-to-face vertex triangulation.
2: Perform sequential Algorithm 4 while the cardinality $|\mathbb{L}| < 4p$
3: Evenly (if possible) divide $\mathbb{L}$ among the p processes $\mathbb{L} = \bigcup_{i=1}^{p} \mathbb{L}^i$, $|\mathbb{L}^p| \approx |\mathbb{L}|/p$
4: $\mathbb{S}^p \leftarrow \varnothing$, $UB(\mathbb{L}^p) \leftarrow \infty$.
5: **while** (all $\mathbb{L}^p$ are not empty: $\mathbb{L}^p \neq \varnothing$) **do**
6: Choose $\mathbb{I}^p \in \mathbb{L}^p$ using selection rule, $\mathbb{L}^p \leftarrow \mathbb{L}^p \backslash \{\mathbb{I}^p\}$
7: $UB(\mathbb{L}^p) \leftarrow \min\{UB(\mathbb{L}^p), \min_{\mathbf{v} \in \mathbb{V}(\mathbb{I}^p)}\{f(\mathbf{v})\}\}$
8: Share $UB(\mathbb{L}^p)$ among processors
9: $\mathbb{S}^p \leftarrow \arg\min \left\{ f(\mathbb{S}^p), \min_{\mathbf{v} \in \mathbb{V}(\mathbb{I}^p)} f(\mathbf{v}) \right\}$
10: Calculate $LB(\mathbb{I}^p)$ using different Lipschitz bounds
11: **if** $(LB(\mathbb{I}^p) < UB(\mathbb{L}^p) - \varepsilon)$ **then**
12: Branch $\mathbb{I}^p$ into 2 simplices: $\mathbb{I}_1^p$, $\mathbb{I}_2^p$
13: $\mathbb{L}^p \leftarrow \mathbb{L}^p \cup \{\mathbb{I}_1^p, \mathbb{I}_2^p\}$
14: **end if**
15: **end while**
16: Collect results $\mathbb{S}^p$

- At initial step, just one process performs sequential branch-and-bound algorithm until the number of unexamined simplices $|\mathbb{L}|$ becomes at least four times bigger than the number of MPI processes. Then these simplices are randomly distributed among all the processes. In this case, the sequential part takes a little longer, but decreases influence of static load balancing.
- The best currently found value of the objective function $UB(\mathbb{L}^p)$ is local, but the processes interchange it in one version of the algorithm whenever a better function value is found.
- The results are collected after the branch-and-bound process finished. The best found solution is a numerical approximation of the global solution.

Computational experiments of both OpenMP algorithm versions were performed on the parallel machine Ness (http://www.epcc.ed.ac.uk/facilities/ness/) at Edinburgh Parallel Computing Center. Ness has a shared-memory architecture which allows users the option to run large threaded jobs (e.g., OpenMP) as well as message-passing 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.

The performance of both MPI algorithm versions were tested on a cluster of personal computers (Vilkas (http://vilkas.vgtu.lt/) cluster at Vilnius Gediminas Technical University). It consists of 14 Intel Core 2 Quad Q6600 (2.4GHz, 4 cores) and 9 Intel Core i7-860 (2.8GHz, 4 cores, 8 threads) computers interconnected via Gigabit Ethernet Switch. Up to 16 Intel Core i7-860 cores have been used in the experiments.

The parallel versions of the algorithms were evaluated using a commonly used criterion of parallel algorithms: speedup $s_p = t_1/t_p$ and efficiency of parallelization $e_p = s_p/p$, where t_p is time used by the algorithm implemented on p processes.

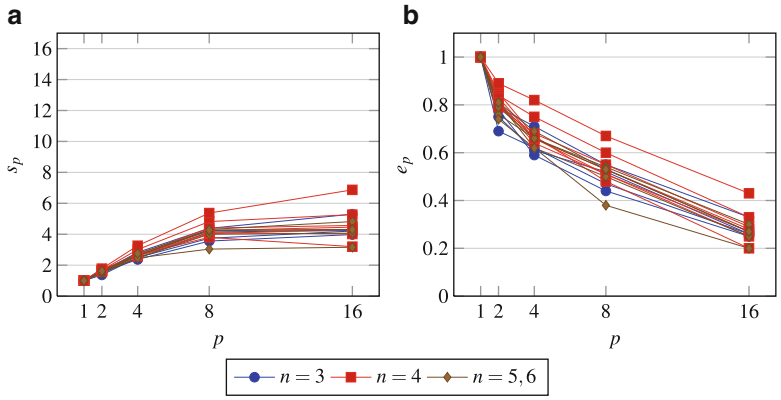

Fig. 2.10 Parallel OpenMP version with vertex verification: (**a**) speedup, (**b**) efficiency

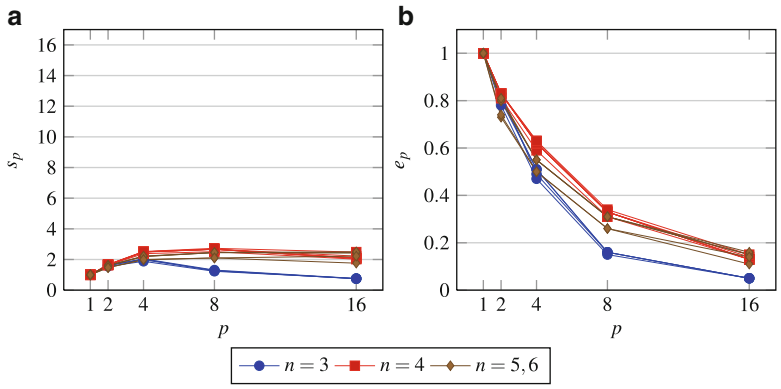

Fig. 2.11 Parallel OpenMP version without vertex verification: (**a**) speedup, (**b**) efficiency

For all parallel versions the aggregate lower bound $\varphi^1 \mu_2^{2,\infty}$ (2.39) with the best first selection strategy was used. Optimization of some problems takes less than a second on a single processor. It is not worthwhile to parallelize the solution of such problems. Therefore in the parallel experiments only more difficult test problems (with a solution time on a single processor of more than 1 s) are used. For test problems $\# = 32, 33$ higher accuracy than in sequential investigations was used.

The diagrams of criteria of parallelization: for various numbers of processes and various dimensionality (n) of test problems are shown in Figs. 2.10–2.13. The diagrams show that the efficiency of parallelization for all parallel versions of algorithm is better for higher dimensional test problems, and much better efficiency is achieved using the MPI versions. The possible reason for this is that significantly less time is spent on communications. The efficiency of OpenMP version with vertex verification is higher than without it.

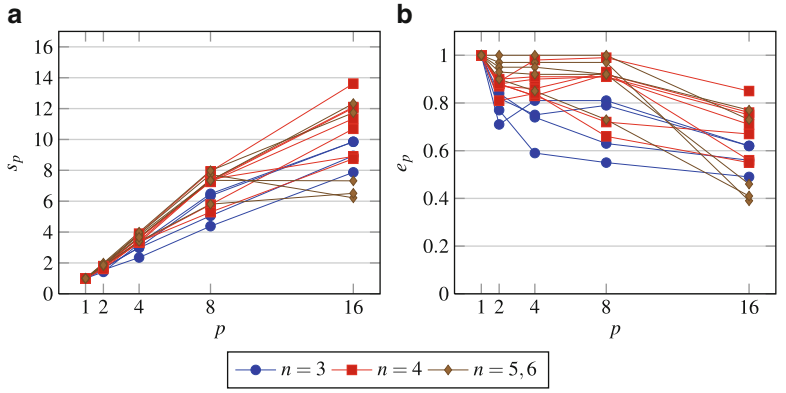

Fig. 2.12 Parallel MPI version without interchange of the best currently found function value: (**a**) speedup, (**b**) efficiency

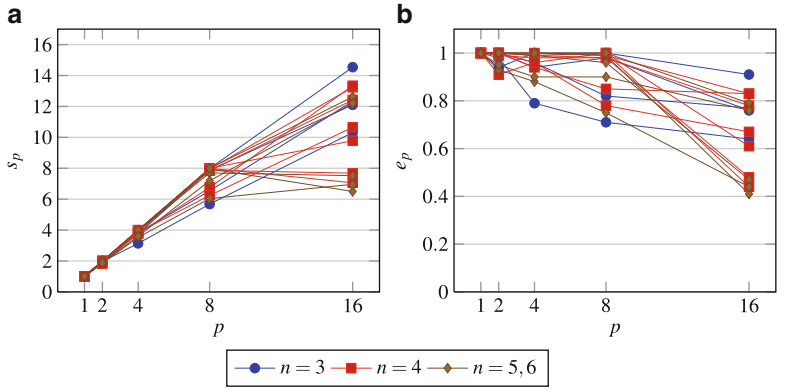

Fig. 2.13 Parallel MPI version with interchange of the best currently found function value: (**a**) speedup, (**b**) efficiency

For both OpenMP parallel versions the number of function evaluations are similar for all numbers of processors and are almost identical to sequential results presented in Table 2.5. For both MPI parallel versions the number of function evaluations using multiple processes are shown in Table 2.7. For most test problems the number of function evaluations using the MPI version with interchange of the best currently found function values (MPI-i) is very similar compared with the MPI version without interchange. For some test problems the number of function evaluations using MPI with interchange is up to 5% smaller than that using MPI without interchange. However, the optimization time using MPI with interchange is almost always longer than for MPI without interchange and is on average about 10% longer (Table 2.8).

Table 2.7 The number of function evaluations using MPI versions on multiple processes

	MPI				MPI-i			
#	$2p.$	$4p.$	$8p.$	$16p.$	$2p.$	$4p.$	$8p.$	$16p.$
14	3423480	3423480	3423480	3423480	3423480	3423480	3423480	3423480
16	3145728	3145728	3145728	3145728	3145728	3145728	3145728	3145728
17	1363119	1363700	1427885	1425216	1363025	1363002	1363010	1363019
20	547622	547622	547981	548079	547622	547622	547622	547622
21	6262233	6262233	6262233	6264548	6262211	6262191	6262224	6262191
22	167217	167217	167217	167217	167217	167217	167217	167217
23	536280	536282	536300	541431	536148	536030	536068	535241
24	988493	988493	988493	988493	988493	988493	988493	988493
25	543712	543716	543717	628371	543451	542736	542984	539629
26	889508	889510	889899	891087	889496	889500	889492	889467
28	314025	314195	316543	318247	313579	313702	313478	313189
29	1916941	1916941	1916941	1916941	1916941	1916941	1916941	1916941
30	6064924	6064924	6064924	6064924	6064924	6064924	6064924	6064924
31	5940304	5940304	5940304	5940304	5940034	5939938	5939645	5938389
32	1488939	1488939	1488939	1488939	1488939	1488939	1488939	1488939
33	11948747	11948747	11948747	11948747	11948747	11948747	11948747	11948747

Table 2.8 Execution time (s) using MPI versions on multiple processes

	MPI				MPI-i			
#	$2p.$	$4p.$	$8p.$	$16p.$	$2p.$	$4p.$	$8p.$	$16p.$
14	5.04	2.63	1.54	0.82	5.77	3.27	1.93	1.09
16	4.92	2.12	1.06	0.63	6.52	2.86	1.43	0.94
17	1.87	1.00	0.47	0.31	2.39	1.31	0.62	0.40
20	0.67	0.42	0.23	0.12	0.86	0.56	0.30	0.17
21	20.37	10.32	4.85	3.28	23.68	12.52	5.65	3.43
22	0.44	0.23	0.11	0.09	0.51	0.26	0.12	0.10
23	1.49	0.75	0.36	0.24	1.78	0.87	0.43	0.26
24	2.96	1.48	0.70	0.45	4.04	1.80	0.83	0.53
25	1.55	0.76	0.37	0.40	1.78	0.88	0.44	0.28
26	2.52	1.35	0.75	0.38	2.90	1.53	0.88	0.48
28	0.99	0.47	0.29	0.17	1.13	0.54	0.34	0.21
29	26.18	13.06	6.66	8.16	27.02	13.55	6.79	8.44
30	80.69	40.57	20.86	21.36	84.67	42.14	21.80	21.87
31	87.29	46.13	26.99	23.40	90.24	47.91	28.04	25.02
32	126.00	63.17	31.58	20.00	125.25	63.24	31.62	21.52
33	1015.47	531.86	267.25	157.45	1040.17	529.17	263.96	157.17

2.8 Experimental Comparison of Selection Strategies

The selection strategies have been introduced in Sect. 1.1. Speed and memory requirements of branch-and-bound algorithms depend on the selection strategy which candidate node to process next [102]. The goal of this section is to experimentally investigate this influence on the performance of sequential and parallel branch-and-bound algorithms. The experiments have been performed solving the same test problems for global optimization as in the previous sections which are described in Appendix A. Branch-and-bound algorithm with simplicial partitions and combination of Lipschitz bounds has been investigated. However, similar results may be expected for other branch-and-bound algorithms.

Parallel experiments have been performed on the computer cluster Ness at Edinburgh Parallel Computing Center (EPCC). It consists of a cluster of two SMP boxes that form the back-end: 2.6 GHz AMD Opteron (AMD64e) processors with 2 GB of memory (32 processors divided into two 16-processors boxes), and a two-processor front-end. The computer cluster runs Linux operating system (Scientific Linux) and Sun Grid Engine.

Let us start the investigation from the sequential branch-and-bound algorithm with simplicial partitions and improved aggregate Lipschitz bound $\varphi^1\psi^2\mu_2^{2,\infty}$ (2.40). The number of function evaluations (fe) and execution time ($t(s)$) using different selection strategies are shown in Table 2.9.

For 2-dimensional test problems the *depth first* selection strategy is the least efficient. For test problems with higher dimensionality $n \geq 3$ the average number of function evaluations are very similar for all selection strategies and the differences are insignificant. For test problems of all dimensionalities the smallest 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. The 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 with the heap structure the insertion is time consuming when the number of elements in the list is large.

The speed of locating the global solution is measured by using the ratio

$$r(f^*) = \frac{fe(f^*)}{fe}, \tag{2.41}$$

where $fe(f^*)$ is the number of function evaluations after which the best global solution f^* is found and fe is the number of function evaluations during the branch-and-bound process. The value of the ratio is between zero and one and shows how fast the global solution f^* is found during the optimization process. The ratio $r(f^*)$ for all test problems is shown in Fig. 2.14. The average ratios $\overline{r(f^*)}$ for test problems of different dimensionalities are shown in Table 2.10. For almost all test problems the smallest ratio is achieved when the *statistical* selection strategy is used and average ratios are more than two times smaller for this strategy comparing with other selection strategies. For test problems of dimensionalities $n = 2$ and $n = 3$ the

Table 2.9 The number of function evaluations and execution time for different selection strategies

	Best first		Depth first		Breadth first		Statistical	
#	fe	$t(s)$	fe	$t(s)$	fe	$t(s)$	fe	$t(s)$
1	716	0.002	4513	0.013	720	0.002	707	0.002
2	185	0.001	199	0.000	185	0.001	184	0.001
3	4843	0.014	5312	0.015	4844	0.014	4843	0.015
4	8	0.000	8	0.000	8	0.000	8	0.000
5	39	0.000	39	0.000	53	0.001	39	0.000
6	1241	0.003	5921	0.016	1241	0.003	1241	0.003
7	12195	0.037	12296	0.034	12195	0.034	12202	0.041
8	341	0.001	462	0.001	342	0.001	340	0.001
9	21724	0.068	21760	0.062	21724	0.058	21724	0.066
10	1495	0.005	1606	0.005	1511	0.004	1493	0.005
11	3598	0.010	5195	0.015	3601	0.009	3979	0.012
12	16938	0.052	17001	0.046	16939	0.046	16937	0.053
13	18737	0.062	18781	0.056	18747	0.055	18766	0.066
mean	**6312**	**0.020**	**7161**	**0.020**	**6316**	**0.018**	**6343**	**0.020**
14	2355490	19.73	2355490	16.90	2355490	17.11	2355490	19.02
15	3784	0.03	6442	0.05	4193	0.03	3797	0.03
16	3145728	27.53	3145728	23.10	3145728	23.15	3145728	27.22
17	1020782	8.58	1059124	7.79	1022743	7.62	1020772	8.50
18	12032	0.09	15025	0.11	12321	0.09	12160	0.10
19	17105	0.14	108820	0.78	17105	0.13	17105	0.13
20	262311	2.09	262340	1.89	262351	1.89	262308	2.07
mean	**973890**	**8.31**	**993281**	**7.23**	**974276**	**7.15**	**973909**	**8.15**
21	5349848	190.17	5350803	185.14	5349944	185.65	5349782	191.10
22	120144	4.13	120144	4.12	120144	4.00	120144	4.15
23	450084	15.73	443780	15.17	443219	15.02	443588	15.74
24	563835	20.02	563836	19.34	563835	19.20	563835	19.96
25	465955	15.81	445906	15.23	443433	15.27	445309	15.91
26	749518	26.27	750735	25.43	750092	25.35	749851	26.26
27	333568	11.79	333471	11.30	333512	11.45	333470	11.63
28	271614	9.38	275474	9.46	276474	9.47	269485	9.35
mean	**1038071**	**36.66**	**1035519**	**35.65**	**1035082**	**35.68**	**1034433**	**36.76**
29	1633849	307.36	1633969	316.83	1633837	310.06	1633837	305.74
30	4590448	864.75	4590448	849.56	4590448	856.56	4590448	879.39
31	5192437	976.94	5202842	976.77	5222690	970.65	5189886	960.17
mean	**3805578**	**716.35**	**3809086**	**714.39**	**3815658**	**712.42**	**3804724**	**715.10**
32	524940	805.94	524940	794.75	524940	779.13	524940	829.04
33	1685793	2451.97	1685793	2462.63	1685793	2464.22	1685793	2578.57
mean	**1105367**	**1324.75**	**1105367**	**1323.92**	**1105367**	**1318.59**	**1105367**	**1374.24**

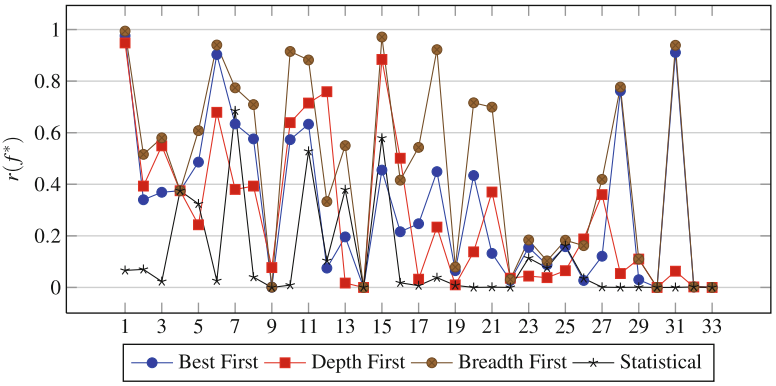

Fig. 2.14 The ratio $r(f^*)$ for the algorithms with different selection strategies

Table 2.10 Average ratio $\overline{r(f^*)}$ for the algorithms with different selection strategies

n	Best first	Depth first	Breadth first	Statistical
2	0.47	0.47	0.63	0.20
3	0.27	0.26	0.52	0.09
4	0.18	0.14	0.32	0.05
5–6	0.19	0.03	0.21	0.00

ratios are very similar for the *best first* and *depth first* search strategies, but for $n \geq 4$ better results (smaller ratio) are achieved when the *depth first* selection strategy is used. For almost all test problems the worst (biggest) ratios are achieved when the *breadth first* selection strategy is used.

The total number of simplices (TNS) and the maximum size of candidate list (MCL) at the search tree for different selection strategies are shown in Table 2.11. For $n = 2, 3$-dimensional test problems (TNS) is largest when the *depth first* selection strategy is used. For higher dimensionality $n \geq 4$ the values of (TNS) are very similar for all selection strategies. But the maximal candidate list (MCL) in the search tree varies significantly depending on selection strategies. The best results (the smallest (MCL)) are achieved when the *depth first* selection strategy is used and it is up to 7,000 times smaller than (MCL) with other selection strategies. The maximal candidate list (MCL) is largest when the *breadth first* selection strategy is used. When the *statistical* selection strategy is used (MCL) is up to $\sim$5 times smaller than when the *best first* strategy is used. This explains why execution time is smaller when *statistical* selection strategy is used. This is because insertion and deletion of candidates to/from heap structure depend on the number of elements in the heap.

Let us continue with investigation of selection strategies in the parallel branch-and-bound algorithm with simplicial partitions and aggregate Lipschitz bound $\varphi^1 \mu_2^{2,\infty}$ (2.39). The average values of speedup $\overline{s_p}$ and efficiency of parallelization $\overline{e_p}$ are shown in Table 2.12. For the test problems of dimensionalities $n = 2$ and $n = 3$ the best average efficiency with various numbers of processes p is achieved when the *breadth first* selection strategy is used. The efficiency of

Table 2.11 The total number of simplices and the maximal size of candidate list

	Best first		Depth first		Breadth first		Statistical	
#	*TNS*	*MCS*	*TNS*	*MCS*	*TNS*	*MCS*	*TNS*	*MCS*
1	1412	216	9024	14	1430	179	1412	161
2	366	53	396	12	368	30	366	29
3	9684	1843	10622	12	9686	1147	9684	303
4	14	3	14	3	14	4	14	4
5	74	10	74	5	74	16	76	6
6	2480	382	11840	14	2480	283	2480	204
7	24388	5489	24590	13	24388	3603	24402	580
8	678	105	922	11	680	63	678	33
9	43446	10444	43518	14	43446	8508	43446	636
10	2984	483	3210	13	2994	341	2984	93
11	7192	1371	10388	14	7194	1037	7956	210
12	33872	5698	34000	13	33874	7258	33872	1268
13	37466	8559	37560	15	37466	8192	37530	2638
mean	**12620**	**2666**	**14320**	**12**	**12623**	**2359**	**12685**	**474**
14	4710974	901316	4710974	24	4710974	689262	4710974	123184
15	7460	761	12878	20	8046	554	7588	102
16	6291450	1543482	6291450	24	6291450	1572864	6291450	368469
17	2041520	378083	2118242	24	2042846	320538	2041538	154576
18	23954	4660	30044	23	24602	3033	24314	2653
19	34204	6321	217634	24	34204	5534	34204	4780
20	524610	111467	524674	21	524652	76043	524610	11060
mean	**1947739**	**420870**	**1986557**	**23**	**1948111**	**381118**	**1947811**	**94975**
21	10699540	1016395	10701582	39	10699674	1656157	10699540	358063
22	240264	28168	240264	35	240264	27865	240264	5170
23	884808	147320	887536	37	885914	159881	887152	132836
24	1127646	164100	1127648	37	1127646	162612	1127646	144028
25	882342	162790	891788	37	885484	157913	890594	130912
26	1498916	216403	1501446	38	1499432	171915	1499678	16513
27	666916	100877	666918	37	666948	98238	666916	8669
28	538946	57926	550924	39	547820	48453	538946	9126
mean	**2067422**	**236747**	**2071013**	**37**	**2069148**	**310379**	**2068842**	**100665**
29	3267554	496255	3267818	132	3267554	398547	3267554	107605
30	9180776	998386	9180776	133	9180776	977293	9180776	159233
31	10379652	1049544	10405564	138	10413428	784132	10379652	136655
mean	**7609327**	**848062**	**7618053**	**134**	**7620586**	**719991**	**7609327**	**134498**
32	1049160	126600	1049160	728	1049160	126720	1049160	69684
33	3370866	249612	3370866	729	3370866	360473	3370866	142415
mean	**4009784**	**188106**	**4012693**	**729**	**4013537**	**243597**	**4009784**	**106050**

Table 2.12 Average speedup and efficiency of parallelization

	2 p.		4 p.		8 p.		16 p.			
n	$\overline{s_p}$	$\overline{e_p}$	$\overline{s_p}$	$\overline{e_p}$	$\overline{s_p}$	$\overline{e_p}$	$\overline{s_p}$	$\overline{e_p}$	mean $\overline{s_p}$	mean $\overline{e_p}$
Best first										
2	1.36	0.68	1.95	0.49	2.79	0.35	4.10	0.26	**2.54**	**0.44**
3	1.86	0.93	2.46	0.61	3.64	0.45	5.13	0.32	**3.27**	**0.58**
4	1.95	0.97	3.65	0.91	6.96	0.87	11.14	0.70	**5.98**	**0.86**
5–6	1.87	0.94	3.64	0.91	6.77	0.85	12.95	0.81	**6.31**	**0.88**
mean	**1.76**	**0.88**	**2.93**	**0.73**	**5.04**	**0.63**	**8.33**	**0.52**		
Depth first										
2	1.61	0.80	1.38	0.35	1.47	0.18	1.39	0.09	**1.47**	**0.36**
3	1.65	0.83	1.92	0.48	2.87	0.36	2.89	0.18	**2.33**	**0.46**
4	1.91	0.96	3.70	0.92	6.80	0.85	9.81	0.61	**5.55**	**0.84**
5–6	1.79	0.89	3.52	0.88	6.69	0.84	12.69	0.79	**6.17**	**0.85**
mean	**1.74**	**0.87**	**2.63**	**0.66**	**4.46**	**0.56**	**6.70**	**0.41**		
Breadth first										
2	1.35	0.68	2.03	0.51	3.04	0.38	4.99	0.31	**2.85**	**0.47**
3	1.87	0.93	3.17	0.79	5.46	0.68	8.26	0.52	**4.69**	**0.73**
4	1.93	0.96	3.74	0.94	7.01	0.88	9.80	0.61	**5.62**	**0.85**
5–6	1.80	0.90	3.58	0.89	6.75	0.84	13.24	0.83	**6.34**	**0.87**
mean	**1.74**	**0.87**	**3.13**	**0.78**	**5.56**	**0.70**	**9.07**	**0.57**		
Statistical										
2	1.30	0.65	1.93	0.48	2.96	0.37	4.14	0.26	**2.58**	**0.44**
3	1.83	0.91	2.50	0.63	3.78	0.47	5.02	0.31	**3.28**	**0.58**
4	1.95	0.98	3.72	0.93	7.24	0.90	10.81	0.68	**5.93**	**0.87**
5–6	1.87	0.94	3.62	0.90	6.85	0.86	13.25	0.83	**6.40**	**0.88**
mean	**1.74**	**0.87**	**2.94**	**0.74**	**5.21**	**0.65**	**8.30**	**0.52**		

parallelization is very similar when the *best first* and *statistical* selection strategies are used. The worst efficiency of parallelization for dimensionalities $n = 2$ and $n = 3$ is experienced when the *depth first* selection strategy is used. For higher dimensionalities $n \geq 4$ the average parallel efficiency is similar for all selection strategies. The efficiency of parallelization decreases less with the same number of processes for difficult (higher-dimensional) test problems compared with simpler test problems.

http://www.springer.com/978-1-4614-9092-0

Simplicial Global Optimization
Paulavičius, R.; Žilinskas, J.
2014, X, 137 p. 51 illus. in color., Softcover
ISBN: 978-1-4614-9092-0