INFORMATICA, 2015, Vol. 26, No. 1, 17–32

DOI: http://dx.doi.org/10.15388/Informatica.2015.36

On the Minimum Number of Simplex Shapes in Longest Edge Bisection Refinement of a Regular n-Simplex

Guillermo APARICIO[1], Leocadio G. CASADO[2]*,
Eligius M.T. HENDRIX[3], Boglárka G.-TÓTH[4], Inmaculada GARCIA[3]

[1]*Research Group TIC146: High Performance Computing – Algorithms University of Almería, Spain*
[2]*Informatics Department, University of Almería (ceiA3), Spain*
[3]*Department of Computer Architecture, Universidad de Málaga, Spain*
[4]*Department of Differential Equations, Budapest University of Technology and Economics Hungary*
e-mail: guillermoaparicio@ual.es, leo@ual.es, eligius@uma.es, igarciaf@uma.es, bog@math.bme.hu

Received: January 2014; accepted: August 2014

Abstract. In several areas like Global Optimization using branch-and-bound methods, the unit n-simplex is refined by bisecting the longest edge such that a binary search tree appears. This process generates simplices belonging to different shape classes. Having less simplex shapes facilitates the prediction of the further workload from a node in the binary tree, because the same shape leads to the same sub-tree. Irregular sub-simplices generated in the refinement process may have more than one longest edge when $n \geqslant 3$. The question is how to choose the longest edge to be bisected such that the number of shape classes is as small as possible. We develop a Branch-and-Bound (B&B) algorithm to find the minimum number of classes in the refinement process. The developed B&B algorithm provides a minimum number of eight classes for a regular 3-simplex. Due to the high computational cost of solving this combinatorial problem, future research focuses on using high performance computing to derive the minimum number of shapes in higher dimensions.

Key words: regular simplex, longest edge bisection, branch-and-bound, combinatorial optimization, simplex shape.

1. Introduction

Global Optimization deals with finding the minimum or maximum value of an objective function f on a closed set with a non-empty interior. We focus here on the so-called standard n-simplex defined in the $(n+1)$-dimensional space:

$$S = \left\{ x \in \mathbb{R}^{n+1} \; \sum_{j=1}^{n+1} x_j = 1; \; x_j \geqslant 0 \right\}, \tag{1}$$

*Corresponding author.

Branch-and-bound methods (B&B) are widely used to solve Global Optimization (GO) problems where the solution is required to have a guaranteed accuracy. Solving problems like non-linear least squares regressions (Žilinskas and Žilinskas, 2013), global optimization of Lipschitz functions (Paulavičius and Žilinskas, 2014; Paulavičius *et al*., 2011), copositivity detection (Žilinskas and Dür, 2011) or blending (Casado *et al*., 2007; Casado *et al*., 2011) make use of this algorithms. The use of simplicial partitioning may be appropriate and efficient in the refinement of the search space (Žilinskas, 2008).

A B&B performs an exhaustive search for optima based on successive decomposition of the search space into smaller sub-regions until a given precision is reached. Bounds on the objective function are calculated for each sub-region, allowing discarding sub-regions where the global optimum cannot be located.

We focus on the binary tree implicitly generated by the branching (refinement) where the simplex division is defined by the Longest Edge Bisection rule (LEB) (Adler, 1983; Horst, 1997; Dickinson, 2014).

Aparicio *et al*. (2013) investigated the effect of the LEB rule on the number of generated simplices, their roundness and whether sub-simplices have similar shapes. A new selection method of the longest edge to be bisected, denoted by LEB_α, was presented. The use of LEB_α selection in the refinement of a 3-simplex produces eight shape classes. One of the goals of this work is to prove that this is the minimum number of shapes that can be achieved.

In a B&B algorithm, the computational cost associated to a node, i.e. the size of its sub-tree, is not known beforehand. In order to get a good prediction, it is desirable to have as few simplex shape classes as possible. If the selected longest edge depends on the shape, sub-simplices with the same shape generate the same type of sub-trees (branches). This will help to predict the pending work and may facilitate dynamic work load balancing for parallel versions of such B&B algorithms (Berenguel *et al*., 2013; Sanjuan-Estrada *et al*., 2011).

The paper is organized as follows. Section 2 describes the simplex refinement process based on the longest edge bisection. Section 3 defines concepts like simplex shape class and simplex quality. Section 4 discusses theoretical aspects on simplices having the same shape. Section 5 shows the B&B algorithm to determine the minimum number of simplex shape classes. Section 6 discusses the findings.

2. Simplex Refinement Using Longest Edge Bisection

In this research, the set to be refined is a regular n-simplex called S_1 scaled to an initial edge length of 1, where the Euclidean norm is used here to calculate the edge length.

$$S_1 = \left\{ x \in \mathbb{R}^{n+1} \sum_{j=1}^{n+1} x_j = \frac{\sqrt{2}}{2};\ x_j \geqslant 0 \right\}. \tag{2}$$

An n-simplex is defined by its set of vertices $V = \{v_1, \ldots, v_{n+1}\}$, $v_j \in \mathbb{R}^{n+1}$. Figure 1 shows a 2-simplex with edge length of 1. The following notation is used:

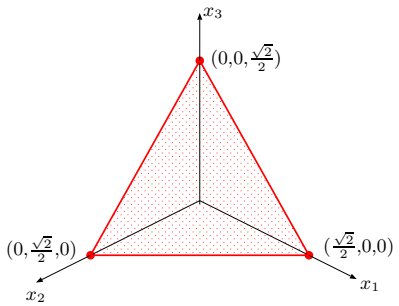

Fig. 1. A regular 2-simplex with edge length 1.

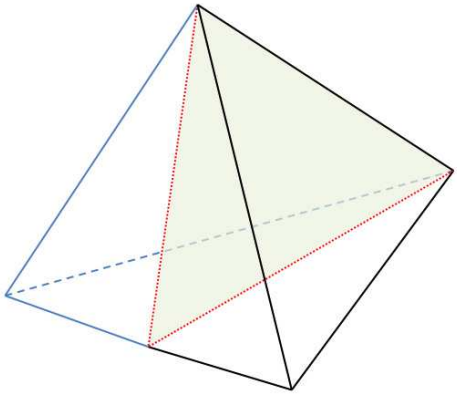

Fig. 2. Longest-edge bisection on a regular 3-simplex.

- $\omega(S)$: the size (width) of a simplex S (longest edge), i.e. $\max_{i,j} \|v_i - v_j\|$.
- $|V|$: the number of vertices.

The longest-edge bisection algorithm is a popular way of iterative refinement in the finite element method, since it is very simple and can easily be applied in higher dimensions (Hannukainen *et al.*, 2014). It is based on splitting a simplex using the hyperplane that connects the mid point of the longest edge of a simplex with the opposite vertices, as illustrated in Fig. 2.

Algorithm 1 provides the Simplex Refinement (SR) algorithm which bisects the initial simplex iteratively. In principle, the refinement can continue indefinitely, but we describe the process here (like in B&B) such that there is a stopping criterion; the branching continues until the size of a node is less than or equal to the desired accuracy ϵ. To study the resulting binary tree, the index i is used to number the simplices. The set Λ provides the leaves of the tree corresponding to the simplices that have not been refined yet. Figure 3 illustrates the result of SR on a 2-simplex with termination criterion $\omega(S) \leqslant 0.5$.

For a 2-simplex, selection among the longest edges in SelectLE is not required as the longest edge is either unique or the choice does not alter the shape of the sub-simplices. To have the procedure SelectLE well defined, the vertices in V should be numbered. For the

Algorithm 1 Simplex Refinement algorithm

Procedure SR(S_1, ϵ)

1: $\Lambda := \{1\}$ ▶ Set of leaf indices; simplices not yet split
2: $ns := 1$ ▶ Number of simplices
3: **while** $\Lambda \neq \emptyset$ **do**
4: Extract a simplex i from Λ
5: **if** $w(S_i) > \epsilon$ **then** ▶ Final accuracy not reached
6: $\{j, k\} :=$ SelectLE(S_i)
7: $\{S_{2i}, S_{2i+1}\} :=$ Bisect(S_i, j, k)
8: Store simplices $2i$ and $2i + 1$ in Λ.
9: $ns := ns + 2$.

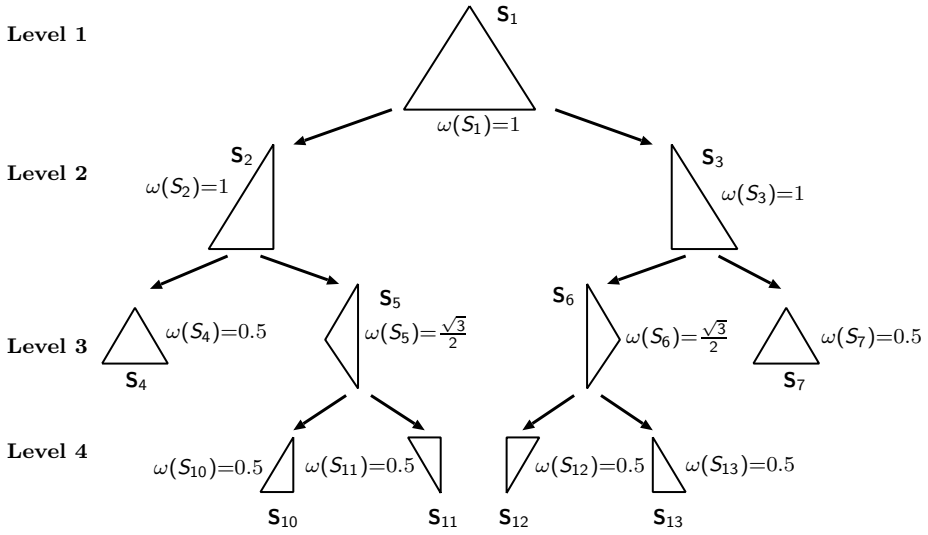

Fig. 3. Binary tree generated by the SR algorithm on a 2-simplex with $\epsilon = 0.5$.

numbering we follow a rule described in Mitchell (1989). He showed that one can avoid edge length calculations in a 2-simplex by always bisecting the edge $\{1, 2\}$ and numbering the new vertex as the last one of the set of vertices in the generated new sub-simplices. The bisection in Algorithm 2 follows the same numbering of the new vertex.

Figure 2 shows the result of the first bisection for a 3-simplex. It does not matter which edge is selected first, because all generated sub-simplices are equal. Notice that there exist three longest edges for the sub-simplices. So, one of the longest edges has to be selected for bisection.

The number of simplices in the finite binary tree of Algorithm 1 depends on how fast the size of the simplices decreases when we go deeper into the tree. Aparicio *et al.* (2013) studies the number and classes of simplices for $n \leqslant 3$ when a specific longest edge selection method, denoted by LEB_α, based on the angles between the edges is used as SelectLE() in Algorithm 1. The question is whether it can be done better, i.e. does

Algorithm 2 Bisection algorithm

Procedure Bisect(S, j, k)
1: Take the vertices v_j and v_k of S to generate $v_{new} := \frac{v_j+v_k}{2}$
2: The new simplices S_l and S_r inherit characteristic of the old: $S_l := S_r := S$
3: Remove v_j from $V \subset S_l$ ▶ Left shift: vertex indices $\in \{1, \ldots, n\}$
4: Number v_{new} as vertex $n+1$ in $V \subset S_l$
5: Remove v_k from $V \subset S_r$ ▶ Left shift: vertex indices $\in \{1, \ldots, n\}$
6: Number v_{new} as vertex $n+1$ in $V \subset S_r$
7: **return** S_l, S_r

a rule exist that generates less classes? We investigate that by solving the combinatorial optimization problem arising when several choices for SelectLE() exist. In order to deal with it, we first define the concept of a simplex shape class and quality measurement.

3. Simplex Shape Classes and Quality Measurement

Definition 1. Two simplices A and B have the *same shape* if each of the vertices of A matches with one of B after scaling, translating and an orthogonal matrix operation that captures rotation and mirroring, see e.g. Plaza and Carey (2002).

Showing that two simplices have the same shape is not easy, because the number of possible matchings is $(n+1)!$. Instead, literature describes various measures aiming at characterizing the so-called shape, roundness or quality of a simplex. It has been shown that most of the measures are equivalent (Parthasarathy *et al.*, 1994; Plaza and Carey, 2002; Dompierre *et al.*, 2005). An example of a description is the following.

Definition 2. (See Dompierre *et al.*, 1998.) A tetrahedron shape measure is a continuous function that evaluates the quality of a tetrahedron. It must be invariant under translation, rotation, reflection and uniform scaling of the tetrahedron, maximum for the regular tetrahedron and minimum for a degenerate tetrahedron. There should be no local maximum other than the global maximum for a regular tetrahedron and there should be no local minimum other than the global minimum for a degenerate tetrahedron.

Aparicio *et al.* (2013) elaborates this idea by defining a simplex quality index, SQ as the ratio between the arithmetic and geometric mean of the set of squared angles between edges of a simplex. Let $\alpha_1, \ldots, \alpha_m$ be the angles between edges of an n-simplex. $SQ(S)$ is defined as:

$$SQ(S) = m \cdot \frac{\sqrt[m]{\alpha_1^2 \cdot \alpha_2^2 \cdot \ldots \cdot \alpha_m^2}}{\alpha_1^2 + \alpha_2^2 + \ldots \alpha_m^2} \in (0, 1], \tag{3}$$

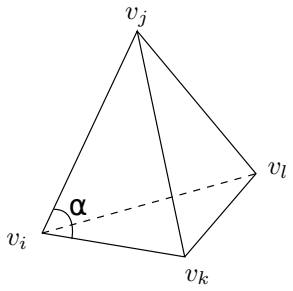

Fig. 4. Obtaining the angle α between two edges sharing a vertex using the corresponding cosine.

where m is the total number of angles determined by

$$m = \frac{n(n+1)(n-1)}{2}. \tag{4}$$

As the length of every single edge linking two vertices in a simplex is known, the angle α between a pair of edges (v_i, v_j) and (v_i, v_k) that share a vertex v_i can be determined by the corresponding cosine. The value of angle α depicted in Fig. 4 can be obtained by calculating

$$\alpha = \arccos\left(\frac{||v_i - v_j||^2 + ||v_i - v_k||^2 - ||v_j - v_k||^2}{2||v_i - v_j||||v_i - v_k||}\right). \tag{5}$$

It can be easily seen that a simplex with very sharp angles has a *SQ* value close to zero and $SQ(S) = 1$ if and only if S is a regular simplex. The idea was elaborated for steering the selection procedure *SelectLE* of the edge to be bisected next.

REMARK 1. Simplices with the same shape have the same set of angles, so they also have the same *SQ* value. That makes having the same *SQ* value a necessary but not a sufficient condition for having the same shape.

REMARK 2. Two simplices with different *SQ* values have different shapes. Therefore, no shape comparison is necessary for a new sub-simplex if there was no simplex with the same *SQ* value so far.

REMARK 3. The computational cost of *SQ* calculation and comparison is far less than to determine if two simplices have the same shape (see Definition 1). This means that one better compares the *SQ* value first.

Although one can generate two simplices with different shape and equal *SQ* value with little effort, it is an open question if simplices with the same *SQ* value exist in the tree generated by the SR algorithm that belong to two different classes. A negative answer would save testing on similarity when two simplices with the same *SQ* value are found.

4. Checking Whether two Simplices Have the Same Shape

There are several ways to check whether simplices fall into the same shape class, according to Definition 1. Specifically, we will look into checking the definition for n-simplices considering their analysis not only in $n+1$ but also in an n dimensional space.

Consider two n-simplices S and Y with corresponding vertex sets V and W. One can also write $S = conv(V)$ and $Y = conv(W)$. In an algorithmic context, V and W are stored as (ordered) $(n+1)$ matrices. In order to follow the similarity definition, it is redefined in Definition 3 in a more exact way using only affine transformations, that is, uniform scaling, translation, rotation and reflection.

Definition 3. Two simplices $S, Y \subset \mathbb{R}^{n+1}$ are shape similar, if there exists a scaling factor $a > 0$, a translation vector $b \in \mathbb{R}^{n+1}$ and an orthogonal rotation/reflection matrix R, such that $\forall w \in Y$, $\exists v \in S$ with $w = b + aRv$.

For the practical check, scaling and translation can be handled by normalizing the simplices S and Y. One can scale both simplices to have a unit longest edge by dividing all the edges by the size of the simplex $\omega(S)$ and $\omega(Y)$ respectively after centering them around the origin. Consider V and W as matrices, then scaling and translation leads to matrices

$$\hat{V} = \frac{1}{\omega(S)}\left(V - \frac{1}{n+1}V\mathbf{1}\mathbf{1}^T\right), \qquad \hat{W} = \frac{1}{\omega(Y)}\left(W - \frac{1}{n+1}W\mathbf{1}\mathbf{1}^T\right), \tag{6}$$

where $\mathbf{1}$ is the all-ones vector. We now show that if simplices $\hat{S} = conv(\hat{V})$ and $\hat{Y} = conv(\hat{W})$ are similar, also S and Y are similar.

Proposition 1. *Let V and W be two nonsingular $(n+1)\times(n+1)$ matrices and $\hat{V}$ and $\hat{W}$ follow from* (6)*. If there exists an orthonormal matrix R such that $\hat{W} = R\hat{V}$ then the simplices $S = conv(V)$ and $Y = conv(W)$ are shape similar.*

Proof. Consider $a = \frac{\omega(Y)}{\omega(S)}$ and $b = \frac{1}{n+1}W\mathbf{1} - \frac{a}{n+1}RV\mathbf{1}$. It should be shown that $\forall w \in Y$, $\exists v \in S$ such that $w = b + aRv$. Any $w \in Y$ can be written as a convex combination of its extreme points: $w = W\lambda$ with $\sum_i \lambda_i = 1$ and $\lambda_i \geqslant 0$, $i = 1, \ldots, n+1$. Now consider $v = V\lambda$, $\hat{v} = \hat{V}\lambda$ and $\hat{w} = \hat{W}\lambda$. Then it follows from the condition and $\sum_i \lambda_i = 1$ that

$$\begin{aligned}
\hat{w} = R\hat{v} \quad &\Rightarrow \quad \omega(Y)\hat{W}\lambda = \omega(Y)R\hat{V}\lambda = \frac{\omega(Y)}{\omega(S)}R\left(V\lambda - \frac{1}{n+1}V\mathbf{1}\right) \\
&\Rightarrow \quad w = W\lambda = \omega(Y)\hat{W}\lambda + \frac{1}{n+1}W\mathbf{1} \\
= \frac{\omega(Y)}{\omega(S)}Rv - \frac{\omega(Y)}{\omega(S)}R\frac{1}{n+1}V\mathbf{1} + \frac{1}{n+1}W\mathbf{1} &= aRv + b. \qquad \square
\end{aligned}$$

As in our reasoning we combine the ideas of vertex sets and a matrix representation, even if $\hat{W} = R\hat{V}$, we still need to find the right ordering of vertices (permutation) in order

to show this is true. A second challenge is that matrix R rotates and reflects around the axis $\mathbb{O} = \{x \in \mathbb{R}^{n+1} \mid x = \alpha \mathbf{1},\ \alpha \in \mathbb{R}\}$ through the origin. The image of R with respect to a matrix V is therefore in the lower dimensional plane $\mathbb{B} = \{x \in \mathbb{R}^{n+1} \mid \sum_i x_i = 0\}$. So $\mathbf{1}$ has to be an eigenvector with eigenvalue 1 of R; $R\mathbf{1} = \mathbf{1}$. The matrices $\hat{V}$ and $\hat{W}$ are singular, as their columns sum to zero. However, for any nonzero scalar α, $\hat{W} + \alpha \mathbf{1}\mathbf{1}^T = R(\hat{V} + \alpha \mathbf{1}\mathbf{1}^T)$, such that R can be found by

$$R_{n+1} = \left(\hat{W} + \alpha \mathbf{1}\mathbf{1}^T\right)\left(\hat{V} + \alpha \mathbf{1}\mathbf{1}^T\right)^{-1}. \tag{7}$$

Even if $\hat{S} = conv(\hat{V})$ and $\hat{Y} = conv(\hat{W})$ are similar, one still has to find the right matching, i.e. the order of the vertices $\{v_1, v_2, \ldots, v_{n+1}\}$ in order to prove it. For any permutation we can compute matrix R_{n+1}; if it is orthonormal, the simplices S and Y are similar.

Instead of working in $(n+1)$-dimensional space, we can also find the rotation-reflection in n-dimensional space. Let $\mathbb{B} = \{x \in \mathbb{R}^{n+1} \mid \sum_i x_i = 0\}$ be the linear space where $\hat{S} = conv(\hat{V})$ and $\hat{Y} = conv(\hat{W})$ can be found and B a matrix containing a basis of $\mathbb{B}$. This means that the simplex $conv(\hat{W})$ is in the space spanned by the columns of B; $conv(\hat{W}) \subset \langle B \rangle$. By considering an orthonormal basis, we can move the analysis to the n-dimensional space.

Proposition 2. *Let B be an orthonormal basis of $\mathbb{B}$, V and W be two nonsingular $(n+1) \times (n+1)$ matrices and $\hat{V}$ and $\hat{W}$ follow from* (6)*. If an $n \times n$ rotation/reflection matrix R ($\det(R) = \pm 1$ and $RR^T = I$) exists such that $B^T \hat{W} = RB^T \hat{V}$, then simplices $conv(V)$ and $conv(W)$ are shape similar.*

Proof. Let Z and X be solutions ($n \times (n+1)$ matrices) of $\hat{W} = BZ$ and $\hat{V} = BX$. The columns are affinely independent and sum to zero; $Z\mathbf{1} = X\mathbf{1} = \mathbf{0}$. An $n \times n$ rotation/reflection matrix R exists such that $Z = RX$. According to the definition, $conv(Z)$ and $conv(X)$ are shape similar in $\mathbb{R}^n$. Multiplication by basis B provides $BZ = \hat{W} = BRX = BRB^T \hat{V}$. The matrix BRB^T is not of full rank; the eigenvalue in the direction $\mathbf{1}$ is zero. Extending the matrices to full rank via $\hat{B} = \left(B, \frac{1}{\sqrt{n+1}}\mathbf{1}\right)$ and $\hat{R} = \left(\begin{smallmatrix} R & \\ 0, \ldots, 0 & \end{smallmatrix} \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}\right)$ provides $\hat{W} = BRX = \hat{B}\hat{R}\hat{B}^T \hat{V}$. Notice that the determinant of $\hat{R}$ is $1 \times \det(R)$. Moreover, $\hat{R}, \hat{B}$ are orthonormal and of full rank: $\hat{B}^{-1} = \hat{B}^T$. We have $\hat{B}\hat{R}\hat{B}^T(\hat{B}\hat{R}\hat{B}^T)^T = \hat{B}\hat{R}\hat{B}^T(\hat{B}\hat{R}^T\hat{B}^T) = I$. So, as $\hat{B}\hat{R}\hat{B}^T$ is an orthonormal rotation-reflection matrix, according to Proposition 1, $conv(V)$ and $conv(W)$ are shape similar. □

A way to construct a orthogonal basis is as follows. For $n = 2$ one can write down such a basis as

$$B_2 = \left(\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \frac{1}{\sqrt{6}}\begin{pmatrix} 1 \\ 1 \\ -2 \end{pmatrix}\right)$$

and recursively define

$$B_n = \begin{pmatrix} B_{n-1} & \frac{1}{\sqrt{n(n+1)}}\begin{pmatrix} \mathbf{1} \\ -n \end{pmatrix} \\ 0, \ldots, 0 & \end{pmatrix}.$$

For instance

$$B_3 = \left(\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \end{pmatrix}, \frac{1}{\sqrt{6}} \begin{pmatrix} 1 \\ 1 \\ -2 \\ 0 \end{pmatrix}, \frac{1}{\sqrt{12}} \begin{pmatrix} 1 \\ 1 \\ 1 \\ -3 \end{pmatrix} \right),$$

$$B_4 = \left(\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \frac{1}{\sqrt{6}} \begin{pmatrix} 1 \\ 1 \\ -2 \\ 0 \\ 0 \end{pmatrix}, \frac{1}{\sqrt{12}} \begin{pmatrix} 1 \\ 1 \\ 1 \\ -3 \\ 0 \end{pmatrix}, \frac{1}{\sqrt{20}} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -4 \end{pmatrix} \right).$$

Proposition 2 shows us that one can also define algorithm SR in a lower dimensional space saving on storage and computational operations. Working with $n+1$ vertices in n-dimensional space, to recover R one can drop one of the vertices and solve similarly to (7). Alternatively, one can look for a least squares approach following Proposition 3.

Proposition 3. *Let Z and X be two $n \times (n+1)$ matrices of rank n. If an $n \times n$ rotation/reflection matrix R ($\det(R) = \pm 1$ and $RR^T = I$) exists such that $Z = RX$, then $R = ZX^T(XX^T)^{-1}$.*

Proof. If $Z = RX$ then $ZX^T(XX^T)^{-1} = RXX^T\left(XX^T\right)^{-1} = R$. □

Now moving back to the data in $n+1$-dimensional space, if we consider in Proposition 3 the matrices $Z = B_n^T \hat{W}$ and $X = B_n^T \hat{V}$ then one can find the n-dimensional rotation matrix by

$$R_n = B_n^T \hat{W} \hat{V}^T B_n \left(B_n^T \hat{V} \hat{V}^T B_n\right)^{-1}. \tag{8}$$

Instead of focusing on the vertex set in matrix representation, one can also look at the n edges of the simplex in n-dimensional space connected to the first (or any) vertex.

Proposition 4. *Let Z and X be two $n \times (n+1)$ matrices with columns being an affinely independent set and F and E $n \times n$ edge matrices defined by columns $f_i = z_{i+1} - z_1$, $i = 1, \ldots, n$ and $e_i = x_{i+1} - x_1$, $i = 1, \ldots, n$. If nonsingular $n \times n$ matrix R exists such that $Z = RX$, then $F = RE$ and $R = FE^{-1}$.*

Proof. Follows from writing out $f_i = z_{i+1} - z_1 = R(x_{i+1} - x_1) = Re_i$. □

Practically Proposition 4 can be used by choosing two matching vertices of $B^T \hat{W}$ and $B^T \hat{V}$ and construct the corresponding edge sets F and E. Computing

$$R_n = FE^{-1} \tag{9}$$

requires less computational operations than (8). The main concern is to find the correct order, i.e. the vertex permutation. Taking this into account, one should find the right permutation matrix P, compute either R_{n+1} from (7) or R_n from (8) or (9) and check whether it is orthonormal. As there are $(n+1)!$ different permutation matrices P, in the worst case R has to be computed and checked $(n+1)!$ times.

5. Finding Selection Rule with the Minimum Number of Shapes

The question is how to generate the binary tree of the Simplex Refinement (SR) algorithm having the minimum number of simplex shape classes. This question is investigated by developing the specific B&B algorithm (MinClassB&B) outlined in Algorithm 3. In the MinClassB&B algorithm, each node corresponds to a tree T rooted at the unit simplex with the following information:

$\Lambda(T)$: The set of leaves of tree T, those nodes needing further processing.
$C(T)$: The set of shape classes found in the nodes of tree T.
$c(S)$: The shape of a simplex S.

Algorithm 3 B&B algorithm to determine the Minimum number of Simplex Classes in the Simplex Refinement process starting from the unit simplex S_1.

Procedure MinClassB&B(S_1)
1: $\Lambda(T_1)$ consists of one leaf, S_1 and $C(T_1) = \{c(S_1)\}$
2: $\Gamma := \{T_1\}$ ▶ Set of trees
3: MinNC $:= \infty$ ▶ Minimum Number of Classes already found
4: **while** $\Gamma \neq \emptyset$ **do**
5: Extract a T from Γ
6: **if** $|C(T)| \leqslant$ MinNC **then**
7: **if** $\Lambda(T) = \emptyset$ **then**
8: MinNC $:= |C(T)|$
9: Store the selection rule *SelectLE* applied to generate T
10: **else**
11: Extract a simplex S_i from $\Lambda(T)$
12: **for** all longest edges (j,k) of S_i **do**
13: Generate new leaves $\{\sigma_1, \sigma_2\} :=$ Bisect(S_i, j, k)
14: **if** $c(\sigma_1) = c(\sigma_2)$ **then**
15: Remove one of the leaves, e.g. σ_2
16: Create a new tree X
17: Leaf set $\Lambda(X) := \{\sigma_l,\ c(\sigma_l) \notin C(T)\} \cup \Lambda(T)$,
18: and shape set $C(X) := \{C(T)\} \cup \{c(\sigma_l) \notin C(T)\}$
19: Store X in Γ
20: Keep track of the chosen edge *SelectLE*(S_i)
21: **return** *MinNC* and the selection rule *SelectLE* to reach it

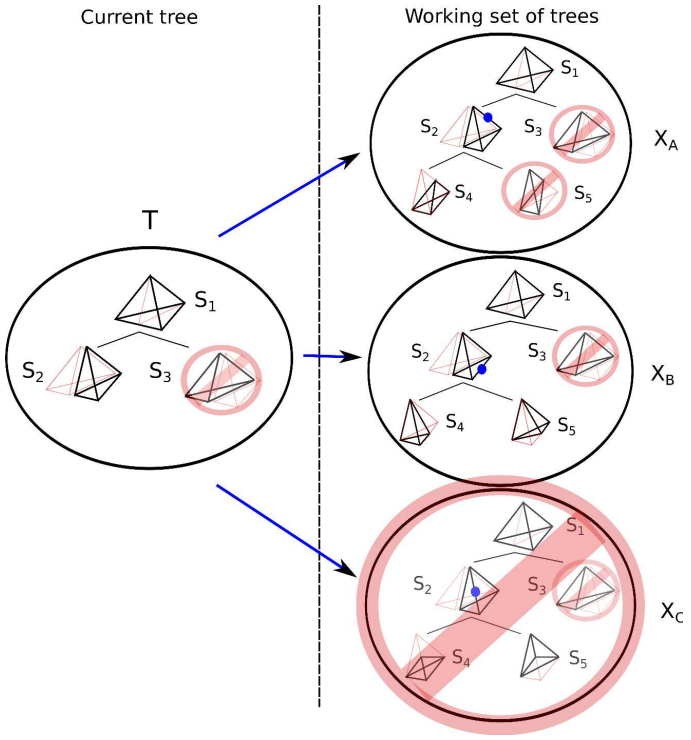

Fig. 5. Generation of new leaves in the *B&B* ToT due to the existence of several longest edges of S_2. X_C is rejected because X_B has the same S_2 descendant shape.

The MinClassB&B algorithm basically builds a tree of binary trees (ToT) with a varying branching degree as illustrated in Fig. 5. The set of leaves of ToT are stored in Γ. The following concepts of branch-and-bound apply.

- MinNC is a global upper bound on the number of classes. Initially, MinNC can be set on a value already found in earlier computations, or infinity otherwise. For instance, an upper bound of eight is already known for a 3-simplex (Aparicio *et al.*, 2013).
- If for a tree T, the number of classes $|C(T)|$ is higher than MinNC, it does not have to be considered anymore. Branching T further will namely always lead to more classes.
- When refinement of T does not lead to new classes, the tree can be removed. This concept is illustrated in Fig. 5.

The global idea is that simplices only have to be stored as leaves in $\Lambda(T)$, if they provide a shape that has not been found before.

The set Γ initially has just one tree T which consists of a set Λ with one leaf, the initial unit simplex. The rest of the algorithm follows the general branch-and-bound structure. Details about which branching is performed first or which tree to be selected first, is left out. The new shape set $C(X)$ is defined in line 18 where $|C(X)|$ is a lower bound of the number of shapes in the tree. In order to reduce the complexity of the pseudocode, it is

Table 1
SelectLE(S_i) in Algorithms 1 and 3 giving the minimum number of classes for a 3-simplex.

Level	SelectLE(S_i)
1	$\{1, 2\}$
2	$\{2, 3\}$
3	$\{1, 2\}$
4	$\{1, 2\}$, for i even $\{1, 3\}$, for i odd
5	$\{1, 2\}$
6	$\{1, 2\}$
7	$\{1, 3\}$, for i even $\{1, 2\}$, for i odd
8	As level 5
9	As level 6
10	As level 7 ...

implicit that a leaf of ToT was first stored in Γ before it is discovered that it can be rejected, when its lower bound is greater than the current upper bound (line 6), or T has no leaves anymore, (line 7). The global upper bound is only updated by the number of shapes of a tree T without leaves in line 8. Setting an initial bound reduces the computation drastically.

Computationally, the hard part of Algorithm 3 is the comparison of the shapes of the new simplices to the existing shape classes in $C(T)$. Moreover, (in line 14) the algorithm checks whether the two children have the same shape; in that case one is removed, as depicted in the T tree of Fig. 5. Additionally, a tree is rejected when its brother tree has produced the same combination of new simplices. This concept is not explicitly described in Algorithm 3 but Fig. 5 shows an example.

6. Results

The MinClassB&B algorithm has been run for the unit 2-simplex and the unit 3-simplex. The 4-simplex case appears computationally very hard due to the large number of possibilities for selecting the longest edge. In the future, we will look into the ability of high performance computing to reach a solution for n-simplices with $n > 3$.

The result for the regular 2-simplex shows the existence of three simplices with different shape as illustrated in Fig. 3. They are generated selecting the longest edge SelectLE(S_i) = $\{1, 2\}$ in the SR process, avoiding edge length calculations.

For the regular 3-simplex, a unique selection rule SelectLE generates the minimum of eight simplex shapes. This confirms the optimality of the longest edge selection (LEB$_\alpha$) studied in (Aparicio *et al.*, 2013). Table 1 describes the SelectLE rule for the minimal shape classes. From level five, the selected LE at each level is repeated every three levels. Notice that i is the index of the simplex S_i which is located at level $L = \lceil \log_2 i \rceil$ in the binary tree generated by SR and the vertices of S_i are reordered following Algorithm 2.

In order to illustrate the optimal SelectLE rule for the 3-simplex i.e. the LEB$_\alpha$ selection rule, Fig. 6 depicts the resulting SR tree where simplices of an already discovered shape

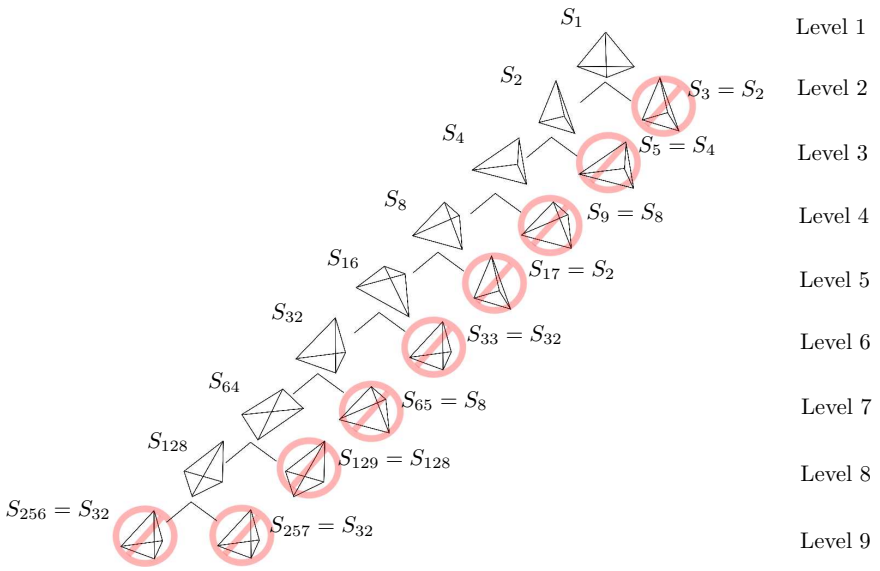

Fig. 6. Binary tree of the SR algorithm without repeating simplices with a shape already found. For a 3-simplex the SelectLE rule of Table 1 provides the same tree as the LEB_α rule.

are not further refined. The LEB_α longest edge selection is not unique for a 4-simplex, thus further investigation is needed.

The behavior of LEB_α longest edge selection applied over an existing B&B algorithm has been tested in (Herrera *et al.*, 2014). The results show that although the LEB_α selection rule is computationally more complex than a rule just selecting the first longest edge, the wall clock time of the algorithm is reduced considerably due to the reduction in the number of evaluated simplices.

7. Conclusions

This work studies the process of iteratively bisecting a regular n-simplex using longest edge bisection. The question is how to select the longest edge to be bisected such that a minimum number of simplex shape classes are generated. To answer this question, a Branch-and-Bound algorithm has been developed that implicitly tests all possible choices for the selection. Numerical results confirm that the eight simplex shape classes for a 3-simplex obtained in (Aparicio *et al.*, 2013) is indeed the minimum number. Numerical determination of the minimum number of simplex classes in larger dimensions requires further investigation of using high performance computing in order to get a result in a reasonable execution time.

Acknowledgements. This work has been funded by grants from the Spanish Ministry (TIN2008-01117 and TIN2012-37483) and Junta de Andalucía (P11-TIC-7176), in part financed by the European Regional Development Fund (ERDF).

References

Adler, A. (1983). On the bisection method for triangles. *Mathematics of Computation*, 40(162), 571–574. doi:10.1090/S0025-5718-1983-0689473-5.

Aparicio, G., Casado, L., Hendrix, E., García, I., G-Tóth, B. (2013). On computational aspects of a regular n-simplex bisection. In: *Eight International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC*, pp. 513–518. doi:10.1109/3PGCIC.2013.88.

Berenguel, J.L., Casado, L.G., García, I., Hendrix, E.M.T. (2013). On estimating workload in interval branch-and-bound global optimization algorithms. *Journal of Global Optimization*, 56(3), 821–844. doi: 10.1007/s10898-011-9771-5.

Casado, L.G., Hendrix, E.M.T., García, I. (2007). Infeasibility spheres for finding robust solutions of blending problems with quadratic constraints. *Journal of Global Optimization*, 39(2), 215–236. doi:10.1007/s10898-007-9157-x.

Casado L. G., García, I., G.-Tóth, B. Hendrix E.M.T. (2011). On determining the cover of a simplex by spheres centered at its vertices. *Journal of Global Optimization*, 50(4), 645–655. doi:10.1007/s10898-010-9524-x.

Dickinson, P.J.C. (2014). On the exhaustivity of simplicial partitioning. *Journal of Global Optimization*, 58(1), 189–203. doi:10.1007/s10898-013-0040-7.

Dompierre, J., Labbé, P., Guibault, F., Camarero, R. (1998). Proposal of benchmarks for 3D unstructured tetrahedral mesh optimization. In: *Proceedings of the 7th International Meshing RoundTable'98*, pp. 459–478.

Dompierre, J., Vallet, M.-G., Labbé, P., Guibault, F. (2005). An analysis of simplex shape measures for anisotropic meshes. *Computer Methods in Applied Mechanics and Engineering*, pp. 4895–4914. doi: 10.1016/j.cma.2004.11.018.

Herrera, J.F.R., Casado, L., Hendrix, E., García, I. (2014). On simplicial longest edge bisection in Lipschitz global optimization. *Computational Science and Its Applications, ICCSA 2014*, *LNCS*, Vol. 8580, pp. 104–114. doi:10.1007/978-3-319-09129-7_8.

Hannukainen, A., Korotov, S., Křížek, M. (2014). On numerical regularity of the face-to-face longest-edge bisection algorithm for tetrahedral partitions. *Science of Computer Programming*, 90, 34–41. doi:10.1016/j.scico.2013.05.002.

Horst, R. (1997). On generalized bisection of n-simplices. *Mathematics of Computation*, 66(218) 691–698. doi:10.1090/S0025-5718-97-00809-0.

Mitchell, W.F. (1989). A comparison of adaptive refinement techniques for elliptic problems. *Transactions on Mathematical Software*, 15(4), 326–347. doi:10.1145/76909.76912.

Sanjuan-Estrada, J.F., Casado, L.G., García, I. (2011). Adaptive parallel interval branch and bound algorithms based on their performance for multicore architectures. *Journal of Supercomputing*, 58, 376–384. doi:10.1007/s11227-011-0594-4.

Parthasarathy, V.N., Graichen, C.M., Hathaway, A.F. (1994). A comparison of tetrahedron quality measures. *Finite Elements in Analysis and Design*, 15(3), 255–261. doi:10.1016/0168-874X(94)90033-7.

Paulavičius, R., Žilinskas, J. (2014). Simplicial Lipschitz optimization without the Lipschitz constant. *Journal of Global Optimization* 59(1), 23–40. doi:10.1007/s10898-013-0089-3.

Paulavičius, R., Žilinskas, J., Grothey, A. (2011). Parallel branch and bound for global optimization with combination of Lipschitz bounds. *Optimization Methods and Software*, 26(3), 487–498. doi:10.1080/10556788.2010.551537.

Plaza, A., Carey, G.F. (2002). Explicación geométrica de una medida de forma de símplices n-dimensional. *Métodos Numéricos para Cálculo y Diseño en Ingeniería*, 18(4), 475–480 (in Spanish).

Žilinskas, J. (2008). Branch and bound with simplicial partitions for global optimization. *Mathematical Modelling and Analysis*, 13(1), 145–159. doi:10.3846/1392-6292.2008.13.145-159.

Žilinskas, J., Dür, M. (2011). Depth-first simplicial partition for copositivity detection, with an application to maxclique. *Optimization Methods and Software*, 26(3), 499–510. doi:10.1080/10556788.2010.544310.

Žilinskas, A., Žilinskas, J. (2013). A hybrid global optimization algorithm for non-linear least squares regression. *Journal of Global Optimization*, 56(2), 265–277. doi:10.1007/s10898-011-9840-9.

G. Aparicio is a telecommunications engineer, informatics PhD student and researcher of the High Performance Computing – Algorithms research group. His research interests include global optimization and parallel computing.

L.G. Casado is associate professor of Informatics Department at University of Almería, Spain from 2001. He obtained his PhD from the University of Málaga. His publications can be found in www.hpca.ual.es/~leo/curr.html. His research interests include, global optimization, parallel computing and secure communications.

E.M.T. Hendrix is a European researcher in the field of optimization algorithms. He is affiliated to the universities of Wageningen and Málaga, but also teaches at other universities. His research interests are global and dynamic optimization and computational impacts.

B. G.-Tóth is associate professor at Budapest University of Technology and Economics. She obtained her PhD from the University of Almería. Her research interest include global and multi-objective optimization, interval analysis and facility location.

I. García. She received the BSc degree in physics from the Complutense University of Madrid, Spain, in 1977, and the PhD degree from the University of Santiago de Compostela, Spain, in 1986. From 1977 to 1987, she was an Assistant Professor, Associate professor during 1987–1997, between 1997 and 2010, Full Professor at the University of Almeria and, since 2010, Full Professor at the University of Málaga. She was head of the Department of Computer Architecture and Electronics at the University of Almería for more than 12 years. During 1994–1995 she was a visiting researcher with the University of Pennsylvania, Philadelphia. Her research interest lies in the field of parallel algorithms for irregular problems related to image processing, global optimization, and matrix computation.

Apie mažiausią simpleksų formų skaičių ilgiausios kraštinės vidurio dalinime pradedant reguliariuoju n-simpleksu

Guillermo APARICIO, Leocadio G. CASADO, Eligius M.T. HENDRIX,
Boglárka G.-TÓTH, Inmaculada GARCIA

Keliose srityse, kaip pavyzdžiui globaliajame optimizavime naudojant šakų ir rėžių metodus, naudojamas simpleksų dalinamas per ilgiausios kraštinės vidurį, kurį galima vaizduoti dvejetainiu medžiu. Dalinimo procesas generuoja skirtingos formos simpleksus. Mažesnis formų skaičius palengvina vėlesnio darbo krūvio prognozavimą dvejetainio medžio mazge, nes tokia pati forma veda prie tokio pačio pomedžio. Net nereguliarūs 3 ar daugiau matmenų simpleksai gali turėti daugiau negu vieną ilgiausią kraštinę. Kyla klausimas, kaip parinkti dalinimui kraštinę, kad simpleksų formų skaičius būtų mažiausias. Mes sukūrėme šakų ir rėžių algoritmą rasti mažiausią formų skaičių. Reguliariam 3 matmenų simpleksui gautas mažiausias formų skaičius yra 8. Didesniam matmenų skaičiui reikalingi lygiagretieji skaičiavimai dėl didelės skaičiavimų apimties.