# Terminal star operations algorithm for tetrahedral mesh improvement

Fernando Balboa, Pedro Rodriguez-Moreno and Maria-Cecilia Rivara

**Abstract** We discuss an innovative, simple and effective Lepp terminal-star algorithm for improving tetrahedral meshes. For each bad quality tetrahedron, one branch of the longest edge propagating path (Lepp) is followed to find an associated terminal star, which is a set of tetrahedra that share a common longest edge (terminal edge). Three alternative improvement mesh operations are considered: simple insertion of the centroid Q of the terminal star, or swapping of the terminal edge, or longest edge bisection. The operation that most improves the mesh is performed whenever significant improvement is achieved. Empirical study shows that, using the dihedral angle quality measure, this simple procedure reduces the bad quality tetrahedra by at least a tenth, with low time cost.

## 1 Introduction

Lepp bisection algorithms and previous longest edge algorithms were designed for local refinement of triangulations in two and three dimensions, for adaptive finite element applications [10, 11, 12, 14]. In three dimensions, for each target tetrahedron t to be refined, the Lepp bisection algorithm follows a longest edge propagating path (Lepp) to find a set of largest (terminal) edges, each of them shared by a set of terminal tetrahedra (terminal star). All the tetrahedra are refined by their longest-

Fernando Balboa
Universidad de Chile, Department of Computer Science, Santiago, 8370456, Chile, e-mail: fernando@balboa.cl
Pedro Rodriguez-Moreno
Universidad del Bio-Bio, Department of Information Systems, Concepcion, 4051381, Chile e-mail: prodrigu@biobio.cl
Maria-Cecilia Rivara
Universidad de Chile, Department of Computer Science, Santiago, 8370456, Chile, e-mail: mcrivara@dcc.uchile.cl

edge (bisection by the plane defined by the midpoint of the longest edge and the two opposite vertices).

In two dimensions the longest edge bisection guarantees the construction of refined triangulations that maintain the quality of the input mesh, improving the average triangle quality [10, 14]; and producing optimal size triangulations [1]. Even when the extension of these properties to three dimensions have not been theoretically proved, empirical evidence shows that the three dimensional algorithm behaves analogously to the two dimensional algorithm in practice, as shown by Rivara and Levin [11]. Applications of these algorithms have been discussed by Williams [19], Jones and Plassmann [7], Castaños and Savage [2], Rivara et al [15].

Lepp Delaunay algorithms for improving two and three dimensional meshes have also been developed [12, 13, 14] which show good practical behavior. Recently, Rivara and Rodriguez-Moreno [17], studied a two dimensional terminal triangles centroid algorithm, proving that the algorithm terminates by producing graded, optimal size, 30 degrees triangulations for planar straight line graph (PSLG) geometries with constrained angles greater than or equal to 30 degrees. The mesh improvement algorithm of this paper generalizes some of the ideas presented in [17] to three dimensions, and is based on some of the ideas presented in [13].

## *1.1 Previous algorithms versus our mesh improvement algorithm.*

Previous algorithms for three dimensional mesh improvement, based on smoothing, swapping, optimization techniques and point insertions have been discussed by Freitag and Ollivier-Gooch [6], Klingner and Shewchuk [8], Dassi et al. [4], Misztal et al. [9]. All these algorithms require good distribution of points inside the input mesh. In what follows we will focus the discussion on the algorithms of these papers. Other algorithms have also been developed such as a sliver exudation algorithm discussed by Edelsbrunner and Guoy [5]; and an algorithm based on optimal Delaunay triangulations developed by Chen and Holst [3].

In a known paper in the mesh generation field, Freitag and Ollivier-Gooch [6], presented a complete empirical study on the use of sequences of three dimensional swapping and smoothing techniques, combined with five alternative (rather complex) optimization strategies. This paper also included a set of recommendations to adequately combine (non trivial) sequences of mesh operations to deal with different mesh improvement issues. Computational testing considers application meshes with good distribution of interior points.

Later, Klingner and Shewchuk [8] discussed a mesh improvement method which combines the mesh operations of Freitag and Ollivier-Gooch, with a set of additional mesh operations (insertion of a new point into a bad quality tetrahedron, two operations for improving boundary tetrahedra, multiphase operations and compound operations) and intensive use of optimization. They succeeded in generating very good meshes (with high computational cost), by assuming that "the spacing of the input vertices in the input mesh is already correct". Based on the operations of Klingner

and Shewchuk, Dassi et al. [4] discussed an optimization method (without using point insertions) to improve the mesh, for geometries formed by unions / intersections of right parallelepipeds with random points in its interior; while Misztal et al. [9] discussed the inclusion of a new multi-face retriangulation operation that can be performed in the boundary of the mesh.

In this paper we propose a simple and efficient mesh improvement algorithm that uses a Lepp path to find a terminal star (set of tetrahedra that share a common longest edge) over which two alternative improvement operations are performed: a new simple centroid insertion operation, or terminal edge swapping, whenever the mesh is locally improved. Our algorithm is more general than previous methods, since this can be also applied to meshes with bad distribution of interior points.

## 2 Lepp bisection and Lepp centroid algorithms

In two dimensions, Lepp(t), the longest edge propagating path of a triangle $t$ [12, 14], is a sequence of increasing triangles that allows for finding a unique local largest edge in the mesh (terminal edge) shared by two terminal triangles (one triangle for a boundary terminal edge). For an illustration see Figure 1 (a). In 3-dimensions Lepp(t) corresponds to a multidirectional searching process [12, 13] that allows for finding a set of terminal edges.

**Definition 1.** $E$ is a terminal edge in a tetrahedral mesh $\tau$ if $E$ is the longest edge of every tetrahedron that shares $E$. In addition, we call terminal star *TS(E)* to the set of tetrahedra that shares a terminal edge $E$.

**Definition 2.** For any tetrahedron $t_0$ in $\tau$, Lepp($t_0$) is recursively defined as follows: (a) Lepp($t_0$) includes every tetrahedron $t$ that shares the longest edge of $t_0$ with $t$, and such that longest edge of $t$ is greater than the longest edge of $t_0$; (b) For any tetrahedron $t_i$ in Lepp($t_0$), this Lepp($t_0$) also contains every tetrahedron $t$ that shares the longest edge of $t_i$ and where longest edge of $t$ is greater than longest edge of $t_i$.

Note that Lepp($t_0$) is a 3D submesh which has a finite and variable number of associated terminal-edges and terminal stars, as illustrated in Figure 1 (b). In the full Lepp bisection algorithm, for each tetrahedron $t$ to be refined, the algorithm computes Lepp($t$) and finds an associated set $W$ of terminal edges. Then for each terminal edge $E$ in $W$, the longest edge bisection of every tetrahedron of the terminal star *TS(E)* is performed, which corresponds to a local refinement operation that maintains a conforming mesh (where the intersection of pair of adjacent tetrahedra is either a common vertex, or a common edge, or a common face). This process is repeated until the target tetrahedron $t$ is refined.

**One path Lepp bisection algorithm**. In this paper we consider algorithms based on partial Lepp computation, following one Lepp branch until one terminal edge is found, according to the following definition:

**Definition 3.** For any tetrahedron $t_0$, of longest edge $L_0$, compute One Branch_Lepp($t_0$) as follows:

a) OneBranch_Lepp($t_0$) includes $t_0$. Then define processing tetrahedron $t_{proc}$ equal to $t_0$ (of longest edge $L_{proc}$).
b) Add to OneBranch_Lepp($t_0$) one tetrahedron $t$ with the greatest longest edge $L_t$, selected between the set of tetrehedra that share edge $L_{proc}$ and having longest edge $L_t$ greater than $L_{proc}$.
c) Repeat for $t_{proc}$ equal to $t$, while there exists a tetrahedron $t$ in step ($b$).

**Remark.** Note that by using Definition 3, in most cases we expect to reach the largest terminal edge in the Lepp submesh.

**Algorithm 1** One Path Lepp Bisection Algorithm ($\tau$, S)

```
Input: τ mesh of tetrahedra; S set of tetrahedra to be refined
Output: refined mesh τ_f
while S ≠ φ do
  For each tetrahedron t_0 ∈ S.
  while t_0 remains in the mesh do
    Compute OneBranch_Lepp(t_0), terminal edge E and terminal star TS(E)
    Perform longest edge bisection of each tetrahedron in terminal star TS(E)
  end while
end while
```

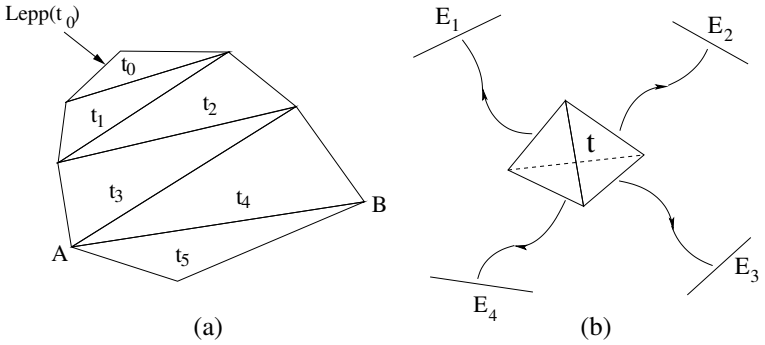


**Fig. 1** (a) Lepp in 2-dimensions: Lepp($t_0$) = $t_0, t_1, t_2, t_3, t_4, t_5$, $AB$ is the terminal edge; (b) Lepp(t) in 3-dimensions has several terminal edges $E_i$

**One path Lepp centroid algorithm**. Here we introduce the simple centroid algorithm in three dimensions. Instead of selecting the terminal edge midpoint, the centroid $Q$ of the terminal star $TS(E)$ is computed. Then $Q$ is simply inserted in the mesh by joining $Q$ with the exterior triangular faces of $TS(E)$, whenever this is a valid three dimensional mesh operation. The algorithm is as follows:

**Algorithm 2** One Path Lepp Centroid Algorithm ($\tau$, S)

```
Input: τ mesh of tetrahedra; S set of tetrahedra to be refined
Output: refined mesh τ_f
while S ≠ φ do
  For each tetrahedron t_0 ∈ S
  while t_0 remains in the mesh do
    Compute OneBrach_Lepp(t_0), terminal edge E and terminal star TS(E)
    Compute centroid Q of terminal star TS(E)
    if simple insertion of Q is valid operation then
      Perform simple insertion of centroid Q
    else
      Perform longest edge bisection of the tetrahedra in TS(E)
    end if
  end while
end while
```

As reported in [11, 13], refinement algorithms based on the bisection of tetrahedra tends to improve the meshes in practice. Experimentation with the Lepp centroid algorithm in the context of this research, shows better behavior than pure longest edge bisection algorithms. However, without considering smart operations, the mesh size increases too much to be competitive for mesh improvement. In the new algorithm of this paper, only smart centroid insertions are performed whenever local mesh improvement is attained.

**Simple centroid insertion**. It is worth noting that the simple centroid insertion is related with smart laplacian smoothing as follows: assume that the terminal edge midpoint $M$ is inserted in the mesh by longest edge bisection of the tetrahedra of the terminal star; then the laplacian smoothing of vertex $M$ is equivalent to the simple insertion of centroid $Q$.

## 3 Swapping of the terminal edge

According to the definition, each terminal star is formed by a variable number of tetrahedra sharing a common longest edge. Consequently, each terminal star is a polyhedron very appropriate for performing edge swapping in a tetrahedral mesh. A full discussion on edge swapping operations can be found in reference [6]. Note that for a set of N tetrahedra sharing an edge, the edge swapping operation replaces the N tetrahedra by a set of 2N-4 tetrahedra. Thus, for N = 3, 4, 5, 6, the associated swapping operations to be used correspond to 3-2, 4-4, 5-6, 6-8 tetrahedra; and so on. In section 5.3 we include a statistical discussion on the practical use of these operations in our improvement algorithm.

## 4 Selective (terminal tetrahedra) centroid / swapping algorithm

The algorithm is formulated in terms of three alternative terminal star operations: simple centroid insertion, terminal star swapping, longest edge bisection (applied to constrained terminal edge). Each one of these operations modifies the interior of the terminal star producing a new tetrahedra set. The algorithm chooses the best operation whenever this locally improves the mesh by using a given factor; otherwise this continues by processing another target tetrahedron.

For each tetrahedra set (tetrahedra in the terminal star, tetrahedra obtained by simple centroid insertion, tetrahedra obtained by terminal edge swapping, tetrahedra obtained by longest edge bisection), we use a function Quality(set) that computes a worst dihedral angle measure as follows. For each tetrahedron t in the set, we compute Dangle as

$$\text{Dangle} = \text{Min}\{\alpha, 180^\circ\text{-}\beta\}$$

where $\alpha$, $\beta$ are the smallest and the largest dihedral angles respectively for tetrahedron $t$, with the conditions $\alpha \leq \theta_1$ and $\beta \geq \theta_2$. Note that $\theta_1, \theta_2$ are user given tolerance parameters for the smallest and largest dihedral angles (see the algorithm 3). Then the Quality(set) function is equal to the smallest Dangle value for the trahedra in the set. The algorithm can be schematically described as follows.

**Algorithm 3** Selective Centroid Swap Algorithm ($\tau$, $\theta_2$, $\theta_2$)

```
Input: tetrahedral mesh τ, θ2, θ2 are tolerances for smallest and largest dihedral angles
Output: improved mesh τf
Find S set of tetrahedra with dihedral angle ≤ θ2 or dihedral angles ≥ θ2.
for each tetrahedron t in S do
  while t remains in mesh do
    compute OneBranch_Lepp(t), terminal edge E, terminal star TS
    compute Qualilty(TS)
    compute centroid Q and Qualilty(insertion)
    (Qualilty is 0 if insertion is not a valid operation)
    compute swapping of TS and Qualilty(Swap)
    if Qualilty(insertion) > Qualilty(Swap) and Qualilty(insertion > Factor * Qualilty(TS))
    then
      Perform insertion of centroid Q
    else
      if Qualilty(swap) > Factor * Qualilty(TS) then
        Perform swapping of the terminal tetrahedra
      else
        set Nothing equal to true
      end if
    end if
    if Nothing is true and Quality(bisection) > Factor * Quality(TS) then
      perform longest edge bisection of TS
    end if
  end while
end for
```

Several remarks are in order:

- We have adjusted the improvement parameter Factor to 1.1. Thus, the local operation is accepted if a 10% of improvement is achieved with respect to the terminal tetrahedra set.
- After one step of the preceding algorithm is performed, certain tetrahedra remain in the mesh, for which the improvement task did not succeed. Then a next step of the algorithm is performed for an actualized set S. The process finishes either if all the tetrahedra in S can not be improved, or after a user fixed number of steps is performed. In practice, after five refinement steps with Factor = 1.1 are performed, no significant mesh improvement is achieved.
- It is worth noting that the algorithm does not include special operations for improving boundary tetrahedra.
- We follow one branch of the Lepp path, according to the Definition 3, to find a terminal star. In practice the results are not different by using alternative Lepp branches.
- We have used a quality criterion that mixes smallest and largest dihedral angles. We use dihedral angles since this is the most used measure in practical mesh generation [7, 8].

## 5 Empirical study

Taking as a basis, three dimensional Lepp-based C++ code, developed by Rodriguez-Moreno [18], our (terminal star operations) improvement algorithm was implemented in a notebook computer with intel core i7 6500U processor without using parallelism of any kind. To study the practical behavior of the algorithm we have considered the complex test problems of Figure 2, and different sets of random points (inside a box).

### *5.1 Comparison of centroid / swap algorithm with only centroid and with only swap techniques*

We compared the full algorithm, which chooses the best (smart) operation between insertion and swapping, against only (smart) insertion and only (smart) swapping. For all the test problems of Figure 2, the full algorithm performs better than the algorithms based on isolated operations. Table 1 illustrates the extreme dihedral angle distribution for the final meshes of the three techniques for the elephant case. Note that, as expected, when only swapping is performed, the final mesh size is smaller than that obtained for the combined Insert / Swap algorithm.

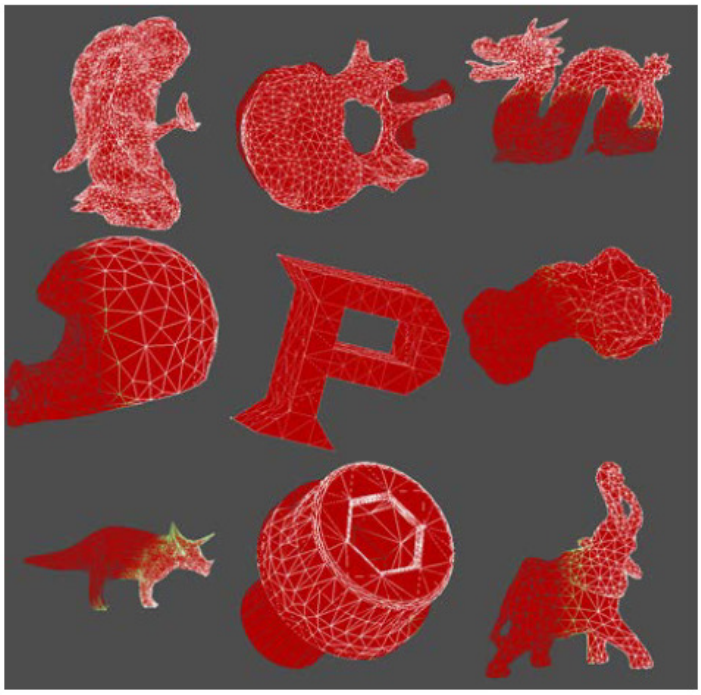

**Fig. 2** Test problems

**Table 1** Comparison of Insert/Swap with only Insert and only Swap. Elephant mesh

| Case | $< 5^\circ$ | $< 10^\circ$ | $< 20^\circ$ | $< 30^\circ$ | $>150^\circ$ | $>160^\circ$ | $>170^\circ$ | Mesh Size |
|---|---|---|---|---|---|---|---|---|
| Initial | 0.08 | 0.54 | 3.9 | 10.51 | 1.6 | 0.54 | 0.07 | 1905 |
| Insert/Swap | 0 | 0 | 0.39 | 3.63 | 0.16 | 0.017 | 0 | 2834 |
| Insert | 0 | 0.05 | 1.38 | 6.54 | 0.48 | 0.09 | 0 | 3092 |
| Swap | 0 | 0.07 | 2.04 | 7.8 | 0.67 | 0.14 | 0 | 1819 |

## *5.2 Improvement performance of the centroid / swap algorithm*

Here we present improvement results for the algorithm of section 4, for the nine test problems of Figure 2 and for one random points mesh. Table 2 summarizes the mesh sizes (initial and final meshes) for these problems, Table 3 includes the refinement time, and Table 4 shows the distribution of the extreme dihedral angles for each one of these problems both for the initial and final meshes. Figure 3 shows the final dihedral angle distributions as compared with the dihedral angle distribution of the initial meshes.

**Table 2** Mesh Sizes.

| Mesh sizes (# tetrahedra) | | | | | | |
|---|---|---|---|---|---|---|
| Mesh | | Retinal | Elephant | P | N090 | Angel |
| Size | Initial | 1374 | 1905 | 926 | 2623 | 13509 |
| | Final | 2663 | 2834 | 1080 | 10014 | 24822 |
| Mesh | | Helmet | Dragon | Spine | Triceratops | Rand2000 |
| Size | Initial | 1268 | 7209 | 3089 | 46202 | 13016 |
| | Final | 2105 | 12104 | 7338 | 85379 | 31030 |

**Table 3** Refinement time.

| Refinement Time (seconds) | | | | | |
|---|---|---|---|---|---|
| | Retinal | Elephant | P | N090 | Angel |
| Time | 5.7 | 1.59 | 0.20 | 18.52 | 63.88 |
| | Helmet | Dragon | Spine | Triceratops | Rand2000 |
| Time | 1.74 | 31.5 | 24.91 | 116.98 | 77.13 |

**Table 4** Distribution of extreme dihedral angle

| %Dihedral Angles | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Mesh | | $<5^\circ$ | $<10^\circ$ | $<20^\circ$ | $<30^\circ$ | $>150^\circ$ | $>160^\circ$ | $>170^\circ$ |
| Retinal | Initial | 0.06 | 1.8 | 10.46 | 22.31 | 3.71 | 0.92 | 0.13 |
| | Final | 0 | 0.06 | 1.87 | 8.12 | 0.99 | 0.12 | 0 |
| Elephant | Initial | 0.008 | 0.54 | 3.9 | 10.51 | 1.6 | 0.54 | 0.07 |
| | Final | 0 | 0 | 0.39 | 3.63 | 0.16 | 0.017 | 0 |
| P | Initial | 0.25 | 0.41 | 1.94 | 5.45 | 0.76 | 0.41 | 0.14 |
| | Final | 0 | 0 | 0.062 | 1.33 | 0.016 | 0 | 0 |
| N090 | Initial | 6.46 | 12.73 | 24.22 | 31.48 | 10.08 | 6.58 | 2.16 |
| | Final | 0.73 | 2.24 | 6.93 | 13.95 | 2.99 | 1.33 | 0.29 |
| Angel | Initial | 0.04 | 0.64 | 5.01 | 12.1 | 1.44 | 0.43 | 0.047 |
| | Final | 0.005 | 0.034 | 0.56 | 4.41 | 0.21 | 0.039 | 0.007 |
| Helmet | Initial | 0.005 | 0.47 | 3.4 | 9.96 | 0.85 | 0.25 | 0.026 |
| | Final | 0 | 0.056 | 0.46 | 4.18 | 0.16 | 0.055 | 0 |
| Dragon | Initial | 0.053 | 0.74 | 5.64 | 13.19 | 1.95 | 0.61 | 0.067 |
| | Final | 0.007 | 0.075 | 0.91 | 5.07 | 0.43 | 0.091 | 0.007 |
| Spine | Initial | 1.56 | 5 | 13.19 | 21.36 | 4.49 | 2.48 | 0.77 |
| | Final | 0.1 | 0.36 | 2.15 | 7.98 | 0.84 | 0.29 | 0.075 |
| Triceratops | Initial | 0.15 | 0.99 | 5.52 | 11.7 | 1.88 | 0.72 | 0.15 |
| | Final | 0.015 | 0.07 | 0.59 | 4.27 | 0.28 | 0.067 | 0.015 |
| Rand2000 | Initial | 1.98 | 4.34 | 10.55 | 18.13 | 4.41 | 2.7 | 1.23 |
| | Final | 0.36 | 0.99 | 2.49 | 6.65 | 1.37 | 0.84 | 0.4 |

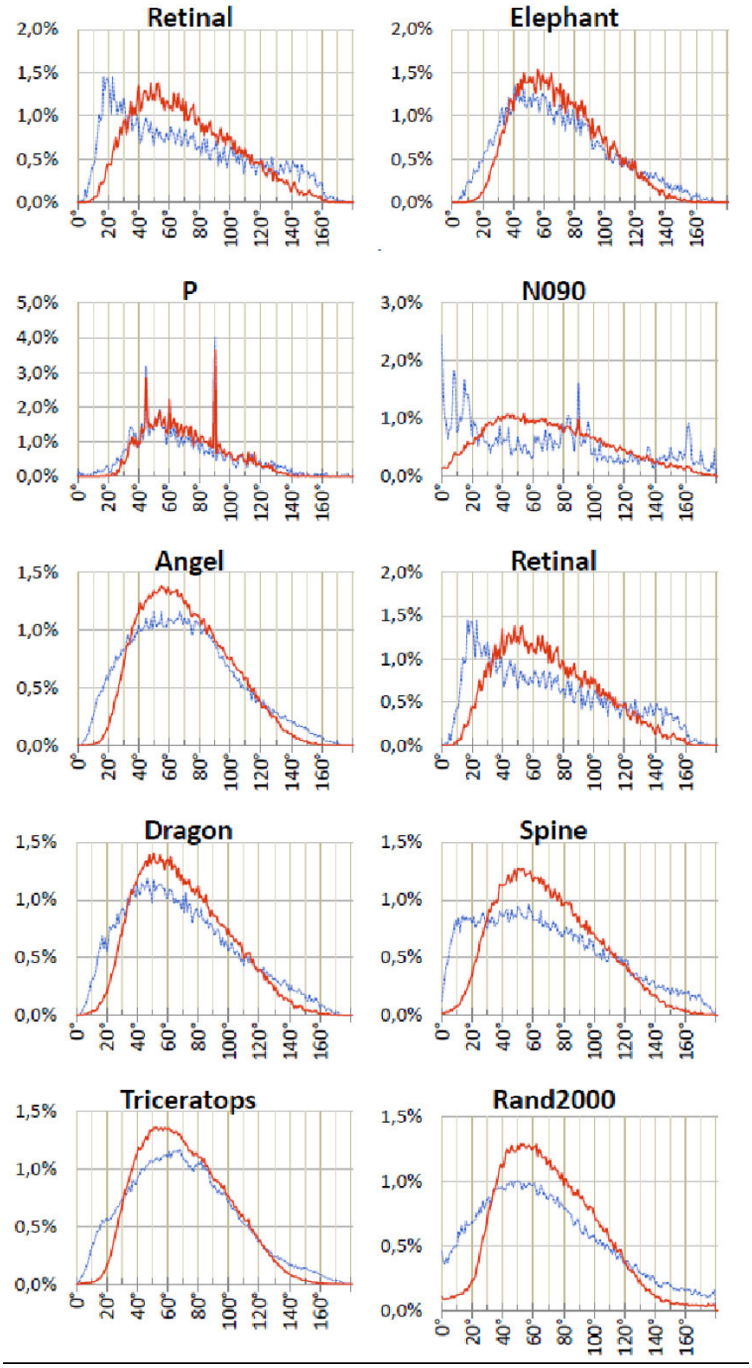


**Fig. 3** Histogram of dihedral angles for the input mesh (in blue) and for the output mesh (in red).

### *5.3 Performance behavior of the terminal star operations*

We have performed a complete statistical study on the frequency of the three operations (centroid insertion, star swapping and nothing) over the terminal stars of different sizes. Firstly, we studied the distribution of the different types of terminal stars (according to their sizes), over all the meshes, finding that the distribution is analogous for all of them. Figure 4 summarizes these results (in average) for all the meshes. Then we studied the percentage of the three operations performed over each type of terminal star by our refinement algorithm. Figure 5 summarizes (in average) these results.

Several remarks are in order:

- 70% of the terminal stars have size $< 6$.
- For most of the stars of size 3, edge swapping is performed.
- The option of doing nothing (to skip the improvement operations) increases with the star size.
- For star sizes $\geq 6$, a small amount of improvement operations is performed.

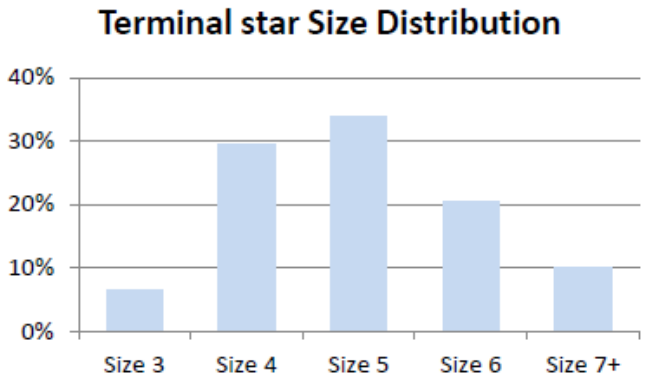


**Fig. 4** Size distribution of the terminal stars (in average for all meshes)

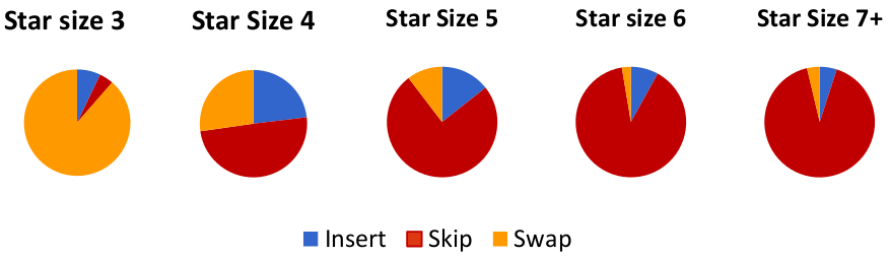


**Fig. 5** Local mesh operations (centroid insertion, star swapping, nothing) as a function of the terminal star size.

## 6 Algorithms comparison

Here we compare our results with those of Klingner and Shewchuk [8], who provided an empirical study (with computing times) for three alternative mesh improvement strategies over a Mac Pro 2.66 GHz Intel Xeon processor, comparable to our hardware. The more recent references [4, 9] did not provide run times. A direct comparison of the computing times shows that our algorithm is 15 to 50 times faster than those of Klingner and Shewchuk. Our algorithm, for a mesh of 12000 tetrahedra takes 31 seconds and for a mesh of 85000 tetrahedra takes 116 seconds. In exchange the methods of reference [8], for a mesh of 11000 tetrahedra take 497 to 940 seconds, and for a mesh of 50000 tetrahedra take 950 to 5823 seconds. Certainly, better quality meshes are obtained by the methods of reference [8], but these can be only applied to input meshes with good distribution of points.

It is worth noting that our improvement algorithm is more general than previous methods, which can be applied to any valid input mesh independently of the distribution of interior points. Furthermore our algorithm always improves the distribution of dihedral angles independently of the meshing problem.

## 7 Concluding remarks

- We have presented an innovative, simple, and effective algorithm for mesh improvement, which works locally over sets of tetrahedra sharing local largest edges in the mesh (terminal stars). This takes advantage of the interesting properties of terminal stars both for point insertion and for terminal edge swapping.
- The method works well both for initial complex meshes having a small number of interior points (elephant, retinal, angel geometries) and for bad meshes having a good distribution of interior points. This is a clear ad-vantage, with respect to previous methods, which require a good distribution of interior points to work [6, 8, 9].

- In our opinion the method has great potential to become the basis of a solid algorithm for three dimensional mesh improvement by adding adequate local operations for improving boundary tetrahedra and intelligent simplification techniques.
- As expected, for the random point meshes, it is more appropriate to use smoothing operations and simplification operations than inserting points.
- The computational cost is low (less than 20 seconds for most problems and 117 seconds for the most difficult retinal geometry). This can be highly de-creased at least in the following two senses: (a) Terminal edge swapping operations are critical and expensive operations, whose implementation can be optimized following the guidelines discussed in reference [6]; (b) The computational work can be constrained to terminal stars of size = 6 according to the analysis of section 3.
- The algorithms are very appropriate for parallelization.
- In a next version of this paper we plan to study the algorithm behavior with other tetrahedra quality measures.

**Acknowledgements** Work partially supported by Departamento de Ciencias de la Computacin, Universidad de Chile, and research Project DIUBB 172115 4/R, Universidad del Bo Bo. We are grateful to the referees who contributed to the improvement of this paper.

## References

1. C. Bedregal, M.C. Rivara, *Longest-edge algorithms for size-optimal refinement of triangulations*. Computer-Aided Des. 46, 246-251 (2014)
2. J. Castaños, J. Savage, *PARED: a framework for the adaptive solution of PDEs. In High Performance Distributed Computing*, 1999. Proceedings The Eighth International Symposium on IEEE, 133-140 (1999)
3. L. Chen and M. Holst *Efficient mesh optimization schemes based on optimal Delaunay triangulations*. Computer Methods in Applied Mechanics and Engineering 200, 967–984 (2011)
4. F. Dassi, L. Kamenski, H. Si, *Tetrahedral mesh improvement using moving mesh smoothing and lazy searching flips*, 25th International Meshing Roundtable, Procedia Engineering, 163, 302-314 (2016)
5. H. Edelsbrunner, D. Guoy, *An experimental study of sliver exudation*. Engineering with Computers, 18, 229-240 (2002)
6. L.A. Freitag, C. Ollivier-Gooch, *Tetrahedral mesh improvement using swapping and smoothing*, Int. J. for Numer. Meth. In Engrg, 40, 3979-4002 (1997)
7. M. Jones, P. Plassmann, *Adaptive refinement of unstructured finite-element meshes*. Finite Elem. Anal. Des., 25, 41-60 (1997)
8. B.M. Klingner, J.R. Shewchuk, *Aggressive tetrahedral mesh improvement*. In Proceedings 16th International Meshing Roundtable, (Springer, Berlin, Heidelberg), 3–23 (2008)
9. M.K. Misztal, J.A. Brentzen, F. Anton, and K. Erleben. *Tetrahedral mesh improvement using multi-face retriangulation*. In Proceedings of the 18th International Meshing Roundtable, B. W. Clark, Ed. Springer Berlin Heidelberg, pp. 539–V555, (2009)
10. M.C. Rivara, *Algorithms for refining triangular grids suitable for adaptive and multigrid techniques*, Int. J. Numer. Meth. in Eng, 20, 745–756 (1984)
11. M.C. Rivara, C. Levin, *A 3-D refinement algorithm suitable for adaptive and multi-grid techniques*, Comm. in Applied Numerical Methods, 8, 281-290 (1992)

12. M.C. Rivara, *New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations*. International Journal for Numerical Methods in Engineering, 40, 3313-3324 (1997)
13. M.C. Rivara, M. Palma, *New LEPP algorithms for quality polygon and volume triangulation: implementation issues and practical behavior*. In Trends in Unstructured Mesh Generation, S.A. Canann and S. Saigal (Eds.). ASME, AMD-Vol. 220, 1-9, 1997
14. M.C. Rivara, *Lepp-bisection algorithms, applications and mathematical properties*. Appl. Numer. Math., 59, 2218-2235 (2009)
15. M.C. Rivara, C. Calderon, A. Fedorov, N. Chrisochoides, *Parallel decoupled terminal-edge bisection method for 3d mesh generation*. Eng. Comput., 22, 111-119 (2009)
16. M.C. Rivara, P. Rodriguez, R. Montenegro, G. Jorquera, *Multithread parallelization of lepp-bisection algorithms*, Appl. Numer. Math., 62, 473-488 (2012)
17. M.C. Rivara, P.A. Rodriguez-Moreno, *Tuned terminal triangles centroid Delaunay algorithm for quality triangulation*. In Proceedings 27th International Meshing Roudtable, Albuquerque, NM USA, 2018.
18. P. Rodriguez-Moreno, *Parallel Lepp-based algorithms for the generation and refinement of 2D and 3D triangulations*, PhD thesis, Department of Computer Science, University of Chile, 160 pages, 2015.
19. R. Williams, *Adaptive parallel meshes with complex geometry*, In Numerical Grid Generation in Computational Fluid Dynamics and related Fields. Elsevier Science Publishers, 201-213 (1991)