A Parallel 2T-LE Algorithm Refinement with MPI

Lorna Figueroa¹, Mauricio Solar^{1,2}, Ma. Cecilia Rivara³, Ma.Clicia Stelling⁴

¹Universidad de Santiago de Chile (USACH), Facultad de Ciencia
Alameda 3363, Santiago, Chile. lorna.figueroa@usach.cl

²Universidad Tecnica Federico Santa Maria (UTFSM)
Avda. Santa Maria 6400, Vitacura, Santiago, Chile. msolar@inf.ut fsm.cl

³Universidad de Chile, Facultad de Matematicas, Santiago, Chile
mcrivara@dcc.uchile.cl

⁴Universidade do Estado do Rio de Janeiro (UERJ), Instituto de Matematica e Estatistica -
CTC, Departamento de Informatica e Ciênci a da Computação.
clicia@ime.uerj.br

Abstract. A description is given of an implementation of a parallel refinement algorithm based on the sequential 2T-LE algorithm of bisection into two triangles (Longest-Edge). The proposed algorithm allows refining in parallel the triangulation of a given geometric mesh. The parallel implementation was carried out on a cluster with a Linux platform communicated through a Message Passing Interface (MPI). The results of the parallel refinement show that as the size of the problem increases, better performance is obtained from the parallel algorithm compared to the sequential algorithm.

Keywords: Parallel model, parallel mesh, parallel refinement algorithm.

1 Introduction

In the analysis of complex scientific and engineering problems the use of geometric mesh models is common. Real applications require operation with geometric meshes with a large volume of data (memory limitations) and with high computing power (HCP) requirements, becoming natural candidates for parallel processing [1]. Existing parallel methods for mesh generation (triangulation) break up the original problem into N subproblems that are processed simultaneously by a set of processors P [2, 3, 4]. Some authors present decoupled solutions, both in mesh generation and in refinement [5, 6]. Decoupling of the mesh subproblems does not require synchronization of communication between the processors when triangulation processes are performed.

In the literature there are various parallel algorithms for solving this problem. Some are aimed at ensuring the quality of the Delaunay refinement [7], or at solutions in unstructured networks [8]. However, these parallel algorithms that are reported solve the problem by decoupling the subproblems, in that way avoiding the synchronization of the boundaries between two processors. Other algorithms deliver a non-conforming solution, which is valid for some situations but not for all cases.

A parallel algorithm can be used to refine all the triangles of a mesh or to refine a set of triangles marked on a geometric mesh. In both cases the quality of the geometric mesh must be maintained. Similarly, the meshes refined in parallel must retain the same quality of the elements and properties as the meshes generated and refined sequentially [6].

In this paper a study is made of the problem of determining the efficiency of a parallel algorithm to refine a set of triangles marked on a geometric mesh. The refinement propagates to neighboring triangles that can be on other processors, and this must be solved efficiently to retain the validity or conformity of the mesh. This communication between processors is done with MPI (Message Passing Interface).

The following section details the problem of refining geometric meshes, and section 3 presents the parallel algorithm for solving the problem. Section 4 shows the testing instances, whose results are analyzed in section 5. Finally, section 6 shows the conclusions.

2 Refinement of a Triangulation

In general, a refinement of a geometric mesh tries to reconstruct good quality triangulations, where geometric criteria are used to define the quality of the triangles, which is related to their equiangularity (how equilateral the triangles are). The following quality criterion is considered in this paper.

- **Definition 1:** A triangulation $\mathfrak{S}_n$ has a certain quality q if its minimum angle is bounded by q . If α is the minimum angle, then $\alpha \geq q$, where $q \leq 18^\circ$.

The most widely used geometric measurement is that of the smallest angle $\alpha(\mathfrak{S}_n)$ in the triangulation. From the definition of quality of a triangulation we get:

- **Definition 2:** An algorithm is said to be q -stable if it retains the mesh's initial quality.

The conformity of a mesh (Figure 1) is defined as:

- **Definition 3:** A mesh is conformal or valid if it is true that the intersection of two neighboring triangles is a side or a common vertex.

Figure 1. (a) Conformal triangulation; (b) Non-conformal triangulation.

Various refining techniques have been proposed to refine triangle meshes in 2D and tetrahedra in 3D. One of these methods is the 2T-LE algorithm, (bisection into two longest-edge triangles, Definition 4), proposed by Rivara [9, 10]. This procedure divides each triangle τ_i of a set of triangles $\mathfrak{S}_n$ by adding an edge from the midpoint v_s of the longest edge to the opposite vertex, creating two new triangles with the same area (Fig. 2). This method propagates the refinement to the neighboring elements of the mesh in such a way that a larger element $\tau_j \notin \mathfrak{S}_n$ is also refined to maintain the mesh's conformity.

- **Definition 4:** 2T-LE divides a triangle τ_i^* of a mesh of triangles $\mathfrak{S}_n$ drawing a line from the midpoint v_s of the longest edge to the opposite vertex, thus forming two new triangles, τ_{i_1} and τ_{i_2} with equal area. This method of bisection propagates to the neighboring triangle in such a way that the conformity of the mesh is maintained.

Figure 2. Partition from the longest side of the initial triangle τ_0

Rivara [9, 10] proves Theorem 1. The result is a conformal mesh, and the transition between large and small elements is smooth.

- **Theorem 1:** Given an input triangulation $\mathfrak{S}_0$ of acceptable quality, which is the discretization of a PSLG (Planar Straight Line Graph) geometry in 2D, the LE algorithms ensure that the triangulations obtained by means of iterative and arbitrary refinement are of the same quality as the input triangulation, in the sense that the smallest angle of each triangulation is greater than or equal to one half of the smallest angle of $\mathfrak{S}_0$.

Based on the empirical study to characterize the performance of the method of bisection from the *longest-edge* [11], the following lemma is formulated:

- **Lemma 1:** Consider a triangulation $\mathfrak{S}_n$ obtained by LE refinement of a triangulation $\mathfrak{S}_{n-1}$ with $n = 1, 2, \dots$. Then the area of the initial triangulation $\mathfrak{S}_n$ covered with almost equilateral triangles increases when n increases. That is, the percentage of almost equilateral triangles also increases.

The algorithms based on LE bisection ensure that:

- The triangulation does not degenerate.
- They guarantee its termination because the edges to be bisected decrease in length over a finite number of steps.
- The final triangulation is conformal.
- As the triangulation is refined overall, a large number of almost equilateral triangles is obtained.

In the implementation of this method, the propagation takes place until non-conformal triangles are no longer produced or until a boundary is reached according to the 2T-LE algorithm described below.

2T-LE Algorithm

Input: $\mathfrak{S}_n^*$ triangulated mesh with marked triangles for refining.

BisectLE(τ_i)

```

    let  $v_p, v_q, v_r$  be the vertices of triangle  $\tau_i^*$ 
    let  $(v_p, v_q)$  be the longest edge of  $\tau_i^*$ 
    let  $v_s$  be the midpoint of  $(v_p, v_q)$ 
    bisect  $\tau_i^*$  on side  $(v_r, v_s)$ , generating 2 new triangles  $\tau_{i_1}$  and  $\tau_{i_2}$ 
    while  $v_s$  is a non-conformal vertex do
        look for a non-conformal triangle  $\tau_j$  adjacent to edge  $(v_p, v_q)$ 
        Bisect LE( $\tau_j^*$ )
    end while

```

Output: $\mathfrak{S}_n$ conformal triangulated mesh.

3 Parallel Model of the 2T-LE Algorithm with MPI

The proposed parallel algorithm consists in breaking up domain Ω_i into μ_i subproblems and assigning each subproblem to a slave processor p_i . In each processor p_i the triangles τ_i marked in its domain are refined and the conflicts at the interfaces are solved with other processors. Once the local refinements have been made, the triangulations $\mathfrak{S}_n$ from the processors p_i (Figura 3) are joined in the master. The master processor uses, beside the 2T-LE refinement algorithm, other algorithms required for the process, such as:

- Reading input files and loading the data structures.
- Sending and coordination of data with the slave processors: it carries out the exchange of messages for sending and receiving data between and from the slaves.
- Generation of output data: it creates the output file with the same structure as the input files.

The slave processors use the following algorithms:

- Receiving the data structures.
- Refinement: it corresponds to the 2T-LE algorithm.
- Conformity: it bisects a non-conformal triangle in an interface, derived from the refinement, leaving it conformal. For this it joins the vertex on the non-conformal side (midpoint of the side) with the opposite vertex.
- Sending and coordination of messages to exchange data with the master.

A scheme of the proposed model is presented in Figure 3, which is a diagram of the work done by the process called Master as well as by the process called Slave; it also shows by means of arrows the communications carried out between both processes. It also shows the synchronization made between the process identified as Master and the processes called Slaves.

(continuation of the graph)

Figure 3. Master-slave algorithm sequence-diagram.

3.1 Message Passing Interface (MPI)

A problem that must be solved is the conflict that arises at the boundaries of the interfaces shared by two slave processors. When a triangle is refined in a slave processor, an interface (boundary with another processor) can be reached and at the same time another neighboring processor will simultaneously reach the same interface refining a triangle in its domain, which shares an edge in common at the interface.

In the parallel refinement model in each partition, there are two types of collisions to be solved: (i) adaptation collisions and (ii) end detection.

- (i) **Adaptation collisions** correspond to the creation of new vertices when bisecting an edge that belongs to an interface, so it is necessary to send a message to the neighboring processor that shares the same affected edge.

The conflict is resolved in two steps:

1. Refinement step: at the start, the master processor sends to each slave processor the partition that it has to process; the latter, during the processing, sends the collision messages to the master processor and continues the refinement of the other triangles marked in its domain.
 2. Conformation step: the master processor collects the collision messages, storing them to be sent to the destination processor once it finishes its local refinement.
- (ii) **End detection** corresponds to the instance in which a slave processor has finished its local refinement, sending a message to the master processor reporting that it finished refining the marked triangles in its partition. The slave waits for an end confirmation from the master processor. If there are conflicts in some of its interfaces, the master sends the collision messages in its interfaces to solve the conformation of the shared triangles. Then it sends again an end message to the master, which collects the end messages of all the slave processors, ending the computation.

The refinement and conformation process is considered ended when all the slave processors have finished, have sent their messages, and received the corresponding end confirmations.

4 Implementation and Test Data of the Parallel Algorithm

The parallel model was implemented in a cluster with 13 Dell PowerEdge 2950 stations with two Intel Dual-Core Xeon 5110 processors at 1.6 GHz, with 4 MB of cache memory, 12-station main memory with 2 GB RAM, and one with 4 GB. A 146-GB (SAS) hard disk and two Gigabit network communications interfaces.

All the stations ran 32-bit Scientific Linux 3.0.8. One station configured with all the services corresponding to a Computing Element GLite/LCG, and Storage Element Glite, and 11 stations configured as Worker Nodes Glite for computing work.

4.1 Test Data

To validate the performance of the implementation of the parallel model three classical grids of the refinement problem were used in a triangulation: Key, S, and Africa. Table 1 shows the characteristics of the test grids. Each grid was refined with three percentages of marked triangles: 10%, 20%, and 30%. Figure 4 shows the image of the grids used.

Table 1: Test Grids

Grid	Triangles	Vertices	Triangles to be refined		
			10%	20%	30%
Key	9533	5000	981	1987	2885
S	9732	5000	986	1971	2964
Africa	9665	5000	965	1909	2872

Figure 4. Grids used (a) Key; (b) Africa; (c) S

4.2 Distribution of Triangles to be Refined per Node

Each test grid of Table 1 was partitioned to distribute the load among the slave processors. The load was distributed into 2, 4, 8 and 16 slaves. As a summary, Figure 5 shows the distribution of the load in the four parallel cases (2, 4, 8 and 16 slave nodes) for the distribution of 10% of the triangles marked on the Key grid. Figure 5 shows that the load distribution is balanced in the different processors not only in terms of the number of triangles of the grid, but also in terms of the marked triangles, which are those that must actually be processed in each slave node.

Figure 5. Distribution of the load of the Key grid with 10% of triangles to be refined.

5 Analysis of Results

5.1 Results with Sequential 2T-LE Algorithm

With the purpose of comparing results, a sequential version of the 2T-LE algorithm was implemented and was executed in the same cluster in which the parallel model was implemented. The results of the sequential execution of each grid with 10%, 20% and 30% of the triangles marked for refining were

obtained. As a summary, Table 2 shows the results for the case of 30% of triangles to be refined in each test grid.

Table 2: Results with sequential 2T-LE algorithm for Key grid

Triangles to be refined	Execution time (s)	Triangles			Vertices	
		Initial	Final	Refine	Initial	Final
10%	120.11	9533	11706	981	5000	6099
20%	714.71	9533	13400	1987	5000	6951
30%	1856.83	9533	14812	2885	5000	7661

5.2 Results with Parallel 2T-LE Algorithm

To validate the implementation of the parallel model of the 2T-LE algorithm, it was executed in the cluster with the load distribution in each case for 2, 4, 8 and 16 slave nodes. The results of the four test grids were obtained with 10%, 20% and 30% of the triangles to be refined. As a summary, Table 3 presents the results of the Key grid with 30% of triangles to be refined in 2, 4, 8 and 16 slave nodes. As a comparison and for the speed up analysis, the results of the sequential algorithm are also shown.

Table 3: Results with parallel algorithm for Key grid with 30% of triangles to be refined

Processors	1	2	4	8	16
Execution Time (s)	1856.83	1474.80	1270.68	1242.72	1182.11
Speed Up		1.26	1.46	1.49	1.57

As seen in Table 3, as the number of processors increases, computation time is reduced, achieving the speed ups shown in the *Speed Up* row. Figure 6 presents the results graphically. It is seen that as the size of the problem increases, the *speed up* of the parallel algorithm improves. That is, the speed up achieved with 30% of triangles to be refined is better than in the case of 20% refinement (of lower computational requirement), and this in turn is better than the case of refining 10% of the triangles.

(a)

(b)

(c)

Figure 6. Results with Key grid with (a) 10%, (b) 20%, and (c) 30% of triangles to be refined.

Figure 7 shows a summary of the speed ups for the case of the Key grid in the three cases of refinement (10%, 20% and 30%) as the number of slave nodes is increased from 2 to 16 (passing by 4 and 8).

Figure 7. Summary of speed up with Key grid with 10% 20% and 30% of triangles to be refined.

Table 4 presents a summary with the average execution times for three of the test grids.

Table 4: Summary of results with the parallel 2T-LE algorithm

Process	Execution time [seconds]								
	Key grid			S grid			Africa grid		
	10%	20%	30%	10%	20%	30%	10%	20%	30%
1	120.11	714.71	1856.83	120.68	760.35	2043.62	98.54	611.52	1789.38
2	89.90	508.30	1264.00	88.08	614.34	1354.79	75.85	441.65	1223.56
4	89.91	503.46	1251.76	87.70	535.00	1335.45	74.34	437.92	1216.00
8	88.10	499.75	1241.04	84.99	522.63	1313.62	73.67	432.11	1192.02
16	82.98	466.13	1182.06	80.05	513.67	1271.13	69.42	408.80	1164.71

6 Conclusions

The results obtained show that the parallelization of the 2T-LE algorithm refines the test grids in a shorter time than the sequential algorithm. The speedups achieved show that as more computation is required in the slave nodes, the parallel algorithm presents increasing performance in relation to the sequential algorithm.

On the other hand, the resultant grids of the refinement maintain their conformity due to the way in which the conflicts are solved in the problem's interfaces. The way of solving the conflicts produces a grid different from that resulting from the sequential algorithm, but the same would happen if the sequential algorithm refines the marked triangles in a different order.

Although in this work there is no space to present all the details, it can be concluded that after carrying out various tests balancing the load in the slave processors, the best results were those with a balanced distribution of the marked triangles to be refined in each slave node, which is what is presented in this paper. It does not depend on the balancing of the triangles in the grid, which will be the subject of another paper to be submitted.

Future work with the parallel model presented is to expand it to the refinement of a 3D grid.

Bibliography

- [1] Chen, Min-Bin; Chiang, Tyng-Ruey; Wu, Jan-Jan. "Parallel 2D Delaunay Triangulations in HPF and MPI". 15th International Parallel and Distributed Processing Symposium (PDPS 2002), San Francisco, California, April 2001, IEEE Computer Society Press.
- [2] Chrisochoides, Nikos. A Survey of Parallel Mesh Generation Methods. Meshing Research Corner. Abril, 2005.

- [3] Chrisochoides, N. and Nave, D. "Parallel Delaunay Mesh Generation Kernel". *Int. J. Numer. Meth. Engng.* 58.161-176. 2003.
- [4] Antonopoulos, Christos D. y otros. "Multigrain Parallel Delaunay Mesh Generation: Challenges and Opportunities for Multithreaded Architectures", Department of Computer Science. College of William and Mary. 2005.
- [5] Linardakis, L., Chrisochoides, N. "Delaunay decoupling method for parallel guaranteed quality planar mesh refinement". *SIAM Journal on Scientific Computing*. Vol. 27, Issue 4 (2006), Pages: 1394 – 1423; 2005, ISSN:1064-8275.
- [6] Rivara, Ma.Cecilia; Calderon, Carlo; Fedorv, Andriy; Chrisochoides, Nikos; "Parallel Decoupled Terminal-Edge Bisection Method For Mesh Generation for 3D mesh generation", *Engineering with Computers*. ISSN 0177-0667 (Print) 1435-5663 (Online). Vol.22,N.2 sept. de 2006.
- [7] Chernikov, A.; Chrisochoides, N.; "Practical and Efficient Point Insertion Scheduling Method for Parallel Guaranteed Quality Delaunay Refinement". *ICS'04*, June 26-July 1, 2004, Malo, France.
- [8] Castaños, J.G.; Savage, J.E. "Parallel Refinement of Unstructured Meshes", *Proceedings of the IASTED International Conference Parallel and Distributed Computing and Systems*, November 3-6, 1999, MIT, Boston, USA.
- [9] Rivara, Maria Cecilia. "Algorithms for refining triangular grids suitable for adaptive and multigrid techniques". *International Journal for Numerical Methods in Engineering*, 20:745-756, 1984.
- [10] Rivara, Maria Cecilia. "Selective refinement/derefinement algorithms for sequences of nested triangulations". *International Journal for Numerical Methods in Engineering*, 28:2889-2906, 1989.
- [11] Figueroa M., Lorna. "Triangulaciones Delaunay y no-Delaunay y particiones de triángulos en base al lado más largo". *Tesis de Magister en Ciencia con mención en Computación*, Universidad de Chile, 2000.