Engineering with Computers (2001) 17: 75–91

Engineering with
Computers

An Algorithm for Three-Dimensional Mesh Generation for Arbitrary Regions with Cracks

J. B. Cavalcante Neto[1], P. A. Wawrzynek[2], M. T. M. Carvalho[1], L. F. Martha[1] and A. R. Ingraffea[2]

[1]Department of Civil Engineering and Computer Graphics Technology Group (Tecgraf), Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil; [2]Department of Civil Engineering and Cornell Fracture Group, Cornell University, Ithaca, NY, USA

Abstract. *An algorithm for generating unstructured tetrahedral meshes of arbitrarily shaped three-dimensional regions is described. The algorithm works for regions without cracks, as well as for regions with one or multiple cracks. The algorithm incorporates aspects of well known meshing procedures, but includes some original steps. It uses an advancing front technique, along with an octree to develop local guidelines for the size of generated elements. The advancing front technique is based on a standard procedure found in the literature, with two additional steps to ensure valid volume mesh generation for virtually any domain. The first additional step is related to the generation of elements only considering the topology of the current front, and the second additional step is a back-tracking procedure with face deletion, to ensure that a mesh can be generated even when problems happen during the advance of the front. To improve mesh quality (as far as element shape is concerned), an* a posteriori *local mesh improvement procedure is used. The performance of the algorithm is evaluated by application to a number of realistically complex, cracked geometries.*

Keywords. Advancing front; Crack analysis; Element shape improvement; Finite elements; Mesh generation; Unstructured mesh

1. Introduction

An algorithm for generating unstructured tetrahedral meshes for arbitrarily shaped three-dimensional regions is described. The algorithm works for regions without cracks, as well as for regions with one or multiple cracks. The cracks may be embedded or surface breaking.

The algorithm was designed to meet four specific requirements. First, the algorithm should produce well shaped elements, avoiding elements with poor aspect ratios. While the algorithm does not guarantee bounds on element aspect ratios, care is taken at each step to generate the best shaped elements possible. Empirical observations, described in Section 3, show that the algorithm is largely successful in meeting this requirement.

The second requirement is that the algorithm generates a mesh that conforms to an existing triangular mesh on the boundary of the region. This is important in the context of crack growth simulation, because it allows remeshing to occur locally in a region near a growing crack. That is, a relatively small number of elements near the crack can be deleted creating a void in the mesh. The crack is extended, and then this algorithm can be used to generate new elements that fill the void, and conform to the elements that were not removed. The remeshed zone is small and localised, leading to fast mesh generation and, for nonlinear problems, minimises the amount of state information that needs to be mapped between an old and new mesh. This requirement matches the successful two-dimensional algorithm previously developed for crack growth simulations [1]. The algorithm, however, is not restricted to small regions near cracks. The examples shown in Sections 3 and 4 demonstrate its use for meshing relatively large regions.

Many of the other meshing algorithms described in the literature generate the mesh on a region's boundary along with the volume mesh. As implied above, for the present algorithm a surface mesh is a required input. The authors do not consider this a significant limitation, however, because there are a number of good surface triangular mesh generators which can be used to generate the required surface mesh (e.g. [2–5]).

Correspondence and offprint requests to: Professor A.R. Ingraffea, Department of Civil and Environmental Engineering, Cornell University, 220 Hollistes Hall, Ithaca, NY 14853-3501, USA. E-mail: ari1@cornell. edu

The third requirement of the algorithm is that it has the ability to transition well between regions with elements of highly varying size. In a crack analysis, it is not uncommon for the elements near the crack front to be two orders of magnitude smaller than other elements in the problem. Some other algorithms work best when all generated elements have similar characteristic size. This is clearly unacceptable for the crack case, and the current algorithm has been designed to have good size transition capabilities.

The fourth requirement is the specific modelling capability for modelling cracks. This requirement arises because cracks are usually idealised as having no volume. That is, the surfaces representing the two sides of a crack face are distinct, but geometrically coincident. This means that nodes on opposite sides of crack faces may have identical coordinates. The algorithm must be able to discriminate between the nodes and select the one on the proper crack face.

The current algorithm incorporates aspects of well known meshing procedures, and includes some original steps. It uses an Advancing Front Technique (AFT), along with an octree to develop local guidelines for the size of generated elements. The AFT is based on a standard procedure found in the literature [6–11], with additional steps to ensure valid volume mesh generation for virtually any domain, including the ones with cracks. Special care is taken during the advancing front procedure to generate elements with the best shape possible.

Although there are many algorithms that have been proposed in the literature in recent years, few of them seem to address all four described requirements in a complete and satisfactory way for crack propagation simulations, such as those undertaken by the authors. One of the main contributions of this work is the consideration of two additional steps that, together with the standard AFT, helps to ensure that the four requirements are satisfied for most of the problems.

The advancing front method is divided into two phases. One, called the geometry-based phase, is based solely on the shape of the elements. In the other phase, called the topology-based phase, the optimal element shape criteria are raised, and the algorithm tries to create valid tetrahedra based only on topology, as in any advancing front method. This two-phase AFT is another main contribution of this work. To improve mesh quality (as far as element shape is concerned), an *a posteriori* local mesh improvement procedure is used.

The additional steps included in the current algorithm, namely back-tracking procedures, are heuristic attempts to avoid the problem of missing closure of the advancing front algorithm. The necessity of these procedures arises from the fact that, unlike triangulation in two dimensions, the discretisation of any given volume into tetrahedra is not formally ensured, unless some additional steps are performed.

Several works in the literature have addressed the solution to this problem recently. For example, the work of Chan and Anastasiou [10] uses local mesh regeneration based on the deletion of sliver tetrahedra in a post-processing step. The *a posteriori* back-tracking procedure of this work has the same objective of that step, but uses a different algorithm. In another recent work, Rassineux [11] also optimises the mesh by reconstruction of sub-volumes that are obtained by the deletion of a group of tetrahedra. It is worth mentioning that, in this work, the back-tracking procedure is not only used in a post-processing stage to improve mesh quality, but also during the advancing front phase, when a sub-volume that cannot be subdivided into tetrahedra is encountered. In this case, not only is the validity of the mesh enforced, but mesh quality is also improved.

Another important difference between the current algorithm and those referred to above is that those algorithms generate internal nodes inside the domain in a prior step, while in this work, internal nodes are generated simultaneously with element generation. Rassineux [11] uses an octree procedure to generate internal nodes prior to the element generation. The current algorithm also uses an octree, but only as a node-spacing function. This approach tends to have a better control over the quality of the generated mesh and, apparently, decreases the amount of heuristic cleaning-up procedures.

Moreover, the current algorithm specifically handles discontinuities in the domain or boundary of the model, such as evolving cracks. The algorithm has recently been used in realistic fracture simulations such as those presented in the work of Carter *et al.* [12].

The body of this paper is divided into three main sections. The following section describes the steps of the algorithm in some detail. Section 3 describes empirical measures of the quality of the elements generated for a number of examples. Section 4 describes empirical measures of the algorithm's asymptotic performance.

2. Description of the Algorithm

The input to the algorithm is a faceted description of the boundary of a region to be meshed. This is

given by a list of nodes defined by their coordinates, and a list of triangular faces defined by their node connectivity. This type of input can represent geometries of any shape, including holes or cracks, and it can be easily incorporated in any finite element system.

The algorithm is organised in the following phases:

I. Octree generation
   a. Initialisation based on boundary mesh.
   b. Refinement to force a maximum cell size.
   c. Refinement to provide minimum size disparity for adjacent cells.
II. Advancing front procedure
   a. Geometry-based element generation.
   b. Topology-based element generation.
   c. Element generation based on back-tracking with face deletion.
III. Local mesh improvement
   a. Laplacian smoothing with validity checks.
   b. Quality evaluation and local back-tracking with element deletion.

2.1. Octree Generation

The primary purpose for the octree is to generate guidelines for the size of the elements generated during the advancing front procedure. The element size distribution through the region is inferred by the size distribution in the input boundary mesh.

The octree generation involves three steps. In the first step, the octree is initialised based on the input data. In the other two steps, the octree is further refined. Figures 1–4 are used to illustrate the process of generating the octree, which for clarity, depict a two-dimensional example using a quadtree. The reader can infer from the two-dimensional example the analogous procedures that take place in three dimensions, which are difficult to depict in a figure. These procedures have been reported in several works in the literature, for example in the work of Shephard and Georges [13]. The point here is not to explain how to generate an octree. This section is included for the sake of algorithm completeness.

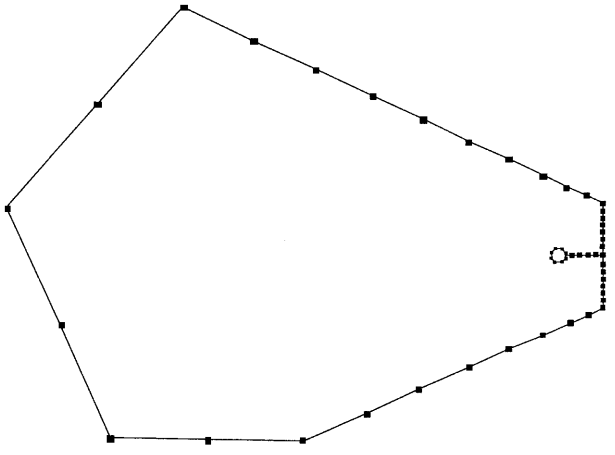

Fig. 1. Hypothetical two-dimensional model and its boundary refinement.

2.1.1. Octree Initialisation Based on Boundary Mesh

Initially, a bounding cube is created based on the maximum range of any of the three Cartesian coordinates of the nodes in the input data. This is the root cell of the octree. Figure 1 illustrates a hypothetical two-dimensional input data represented by the model and its boundary refinement. This model has an edge crack on its right-hand side. At the crack tip, the boundary is contracted as if it had specially placed crack-tip elements. The boundary model presents an increasing degree of refinement from the left side to the right side.

In the first step of the octree generation, represented by the initialisation of the octree, each face of the input boundary mesh is used to determine the local depth of subdivision. The octree cell containing the centroid of each input face is determined. If the area of this cell's face[1] is larger than the area of the boundary face, then this cell is subdivided into eight smaller cells. This process is repeated recursively, and finishes when the area of the cell's face is smaller than a constant times the area of the boundary face. In this implementation, a factor of 0.4 was used. The use of this factor is recommended by other work found in the literature [13] to avoid excessive refinement in the octree generation. This process is repeated for all faces of the input data. The results are illustrated for the two-dimensional example in Fig. 2.

2.1.2. Octree Refinement

The previous step can leave large octree cells in the interior of the region. In a second step, the octree is refined to guarantee that no cell in the interior is bigger than the largest cell at the boundary. This will avoid excessively large elements in the domain interior. Figure 3 shows the resulting quadtree after this operation for the two-dimensional example.

The octree is further processed in a third step, to force only one level of refinement between neigh-

[1] Each cell is a cube that has equal six faces, so any face can be used.

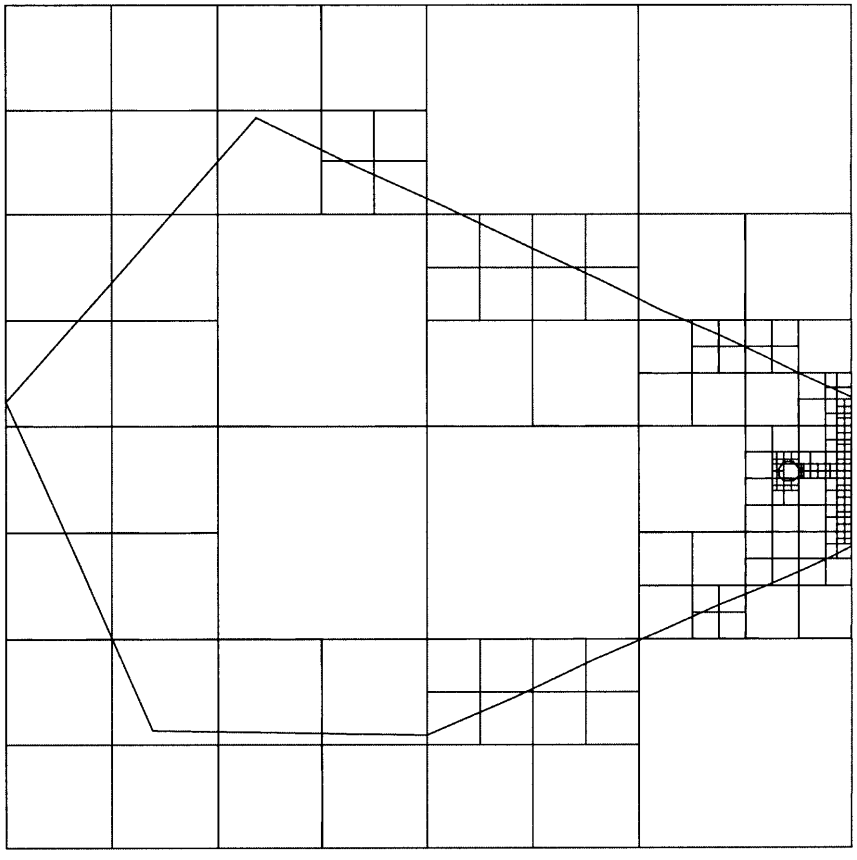

Fig. 2. Initial quadtree of two-dimensional model.

bouring cells. This enforces a natural transition between regions of different degrees of refinement. This operation is performed by traversing the octree and examining the level of refinement between adjacent cells. If the difference is more than one level, the appropriate cells are refined until the criterion is satisfied. Figure 4 shows the quadtree generated for the two-dimensional example after this procedure.

2.2. Advancing Front Technique

The advancing front technique starts with a surface that bounds a region. Volume elements are 'extracted' or 'pared' from the region one at a time. As each element is extracted, the bounding surface is updated and the process is repeated. The procedure terminates when the entire region is meshed, or when one or more internal unmeshed cavities remain, from which valid elements cannot be extracted.

In the present algorithm, the advancing front process is divided into three phases to ensure generation of valid volume meshes. In the first phase, a geometry-based element generation is pursued to generate elements of optimal shape. After this ideal phase is exhausted, and no more optimal elements can be generated, a topology-based element generation takes place, trying to create valid, but not necessarily well shaped, elements in the remaining region. In the last phase, a back-tracking procedure is used to delete element faces that are preventing the algorithm from completing a mesh.

2.2.1. Geometry-Based Element Generation

Ideally, the entire mesh would be generated in the geometry-based phase. However, this depends upon the geometry and topology of the given boundary model and, from our experience with the present algorithm, is strongly related to the shape quality of the given boundary mesh.

Boundary contraction lists. The process starts with the creation of the initial advancing front, which is formed by the given boundary mesh. The current boundary mesh is stored in two separate lists. The first is a list of active faces, which includes all boundary mesh faces that have not been used in an attempt to generate valid tetrahedra. The other is a

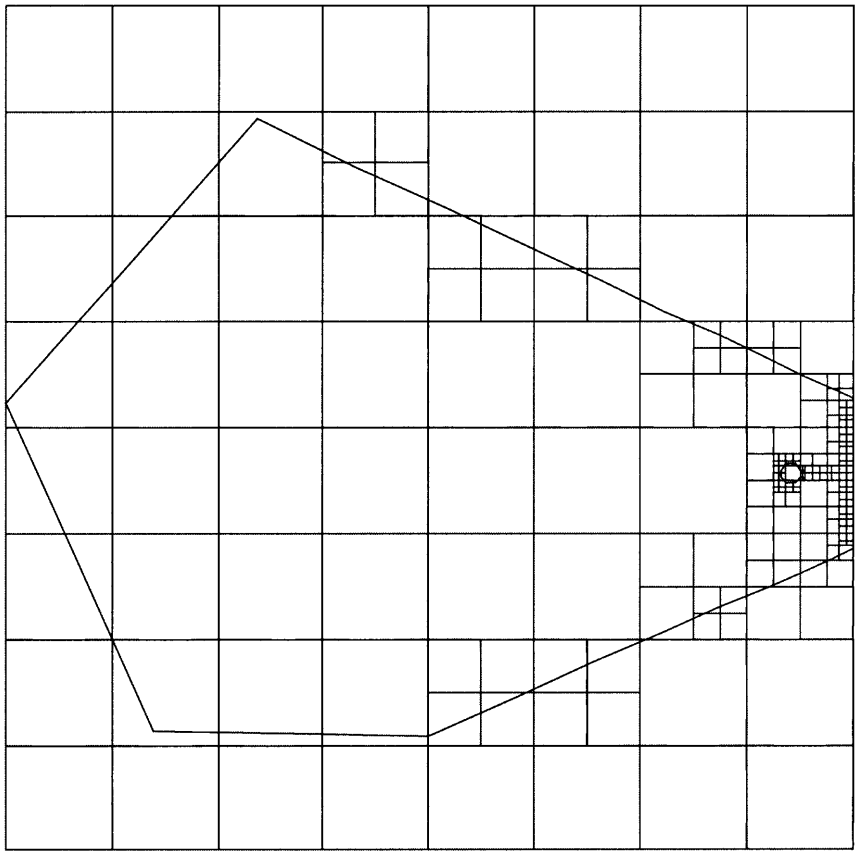

Fig. 3. Quadtree of two-dimensional model after forcing biggest cell size at boundary.

list of rejected faces, that is the faces that failed in the generation of elements for the current phase. Initially, all faces of the given boundary mesh are stored in the first list, which is the list used in the geometry-based generation phase.

The list of active faces is sorted by the area of the faces. This ensures that the first face selected from this list will always be the face with the smallest area. This has been recommended by other authors [6] to prevent large elements from penetrating into regions with small area faces.

It was also found convenient for some phases in the algorithm to have an additional data structure that holds a list of adjacent boundary faces for each node on the current advancing front. This data structure is initialised for all the nodes of the given boundary mesh. The data structure is updated as the boundary contraction procedure progresses.

Generation of optimal elements. In the geometry-based element generation phase, the current boundary mesh advances by trying to form tetrahedra based mainly on geometrical considerations. At each step, a triangular boundary face, referred to as the *base face*, is chosen from the list of active faces. This face has the smallest area in the current front, and its normal points to the interior of the region to be meshed. The current front is represented by all existing faces at a certain time of the algorithm.

The procedure for generating a tetrahedron in this phase is explained by means of Fig. 5. This procedure is divided into the following steps:

- The optimal location *N1* for the vertex of the tetrahedron to be formed is determined with the help of the octree. The octree cell containing the centroid *M* of the base face is determined. The optimal point *N1* lies on a line perpendicular to the base face passing through this centroid. The distance from the optimal point to the base face centroid is equal to the octree cell size.
- The optimal point defines a search region where the vertex of the new tetrahedron may be located. This region is a sector of a sphere whose centre is the optimal point, and whose radius is proportional to the octree cell size. In the current implementation, a proportionality constant of 1.0 was adopted. This sphere defines an upper bound

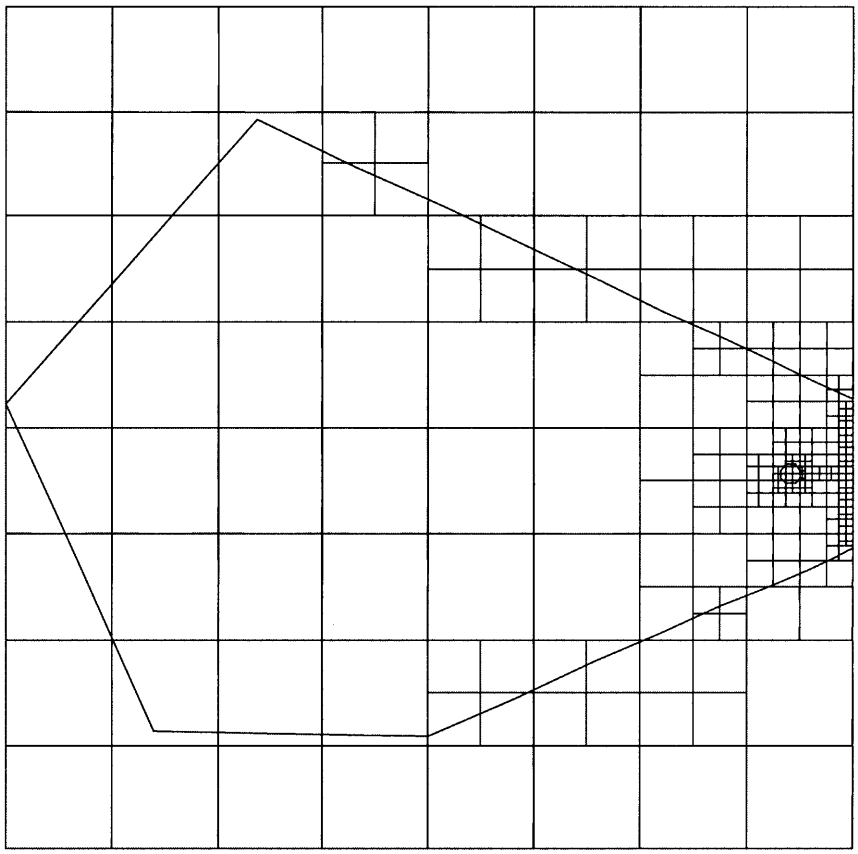

Fig. 4. Quadtree of two-dimensional model after forcing maximum of one level of difference.

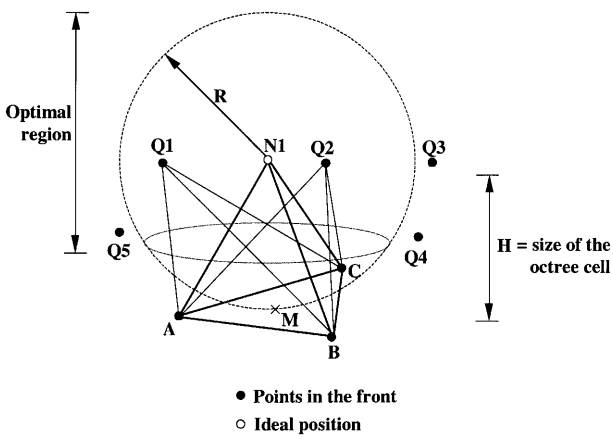

Fig. 5. The determination of a tetrahedron in three-dimensions.

for the distance between the target vertex of the tetrahedron and the centroid of the base face. A lower bound is also defined to ensure that the generated tetrahedron will have volume greater than the smallest acceptable volume. In the current implementation, this lower bound is defined by a tetrahedron with height equal to 1/10 of the distance between *N1* and *M*. The optimal region is used for two reasons. First, to ensure shape quality of the elements to be generated, and secondly, to ensure that new internal nodes will be created only when it is strictly necessary, and always in good positions. Figure 5 shows, based on the bounds described, that points *Q1* and *Q2* are acceptable for forming a new tetrahedron, while points *Q3*, *Q4* and *Q5* are not.

- If no existing node is inside the optimal region, a new node is inserted at the optimal location *N1* and an element is generated using this node. If only one node exists in the region, this node is used to generate the element. If more than one node is found in the region, they are ranked according to the solid angle that they will create with the base face. The node that will create the largest solid angle is used to generate the element. The solid angle is evaluated by projecting the base face onto a unit sphere with centre at vertex *i* and computing the area of the spherical polygon thus determined [14], as shown in Fig. 6.
- Additional geometric checks are performed to ensure that the faces of the new element do not intersect any existing face of the advancing front. If this is the case, the element is rejected.

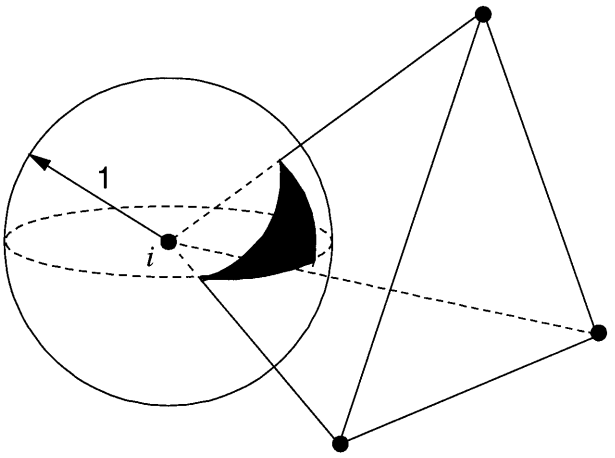

Fig. 6. The definition of a solid angle for a vertex *i* of a tetrahedron.

- For crack problems there might be two or more nodes with the same coordinates. Figure 7 illustrates this case for a two-dimensional example at the left-hand side. This figure also shows a three-dimensional situation, at the right-hand side. The algorithm selects the proper node using a simple test which is based on the lists of adjacent boundary faces of the nodes on the advancing front. When two candidate nodes at the crack surface are selected to form an element, the node which lies on the same side of the base face with respect to the crack surface is chosen. The normals to the crack faces adjacent to the selected nodes are used to perform this test. This check assumes that all crack surfaces are smooth.
- Once a valid tetrahedron is generated for the current base face, the list of active faces is updated. This is done through the following steps.

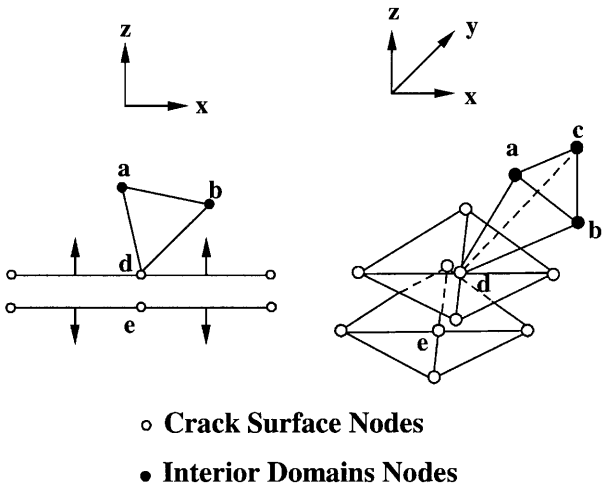

Fig. 7. Selection of a candidate node at a crack surface.

First, the base face is removed from the list. Then, for the other faces of the element: each face is deleted if it coincides with a face already in the list, or the face is inserted in the list as a new one. The insertion is done maintaining the ordering of the list of active faces according to face area.
- Due to geometric bounds imposed by the current front, there are situations in which the algorithm fails to form a valid tetrahedron for the current base face. In these cases, the current base face is removed from the list of active faces and is stored in the separate list of rejected faces. It might happen that a face is subsequently removed from this latter list if it is used to form part of a valid tetrahedron for an adjacent base face.
- When there are no more faces in the list of active faces, the algorithm tries to generate elements using the faces that were rejected previously. Some base faces that failed previously might now work because the front has changed with the addition of elements. A similar procedure is mentioned in the literature [9]. The geometry-based element generation phase ends when either there are no faces left in the boundary contraction lists (in which case an optimal mesh was generated), or when a rejected face fails a second time.

2.2.2. Topology-Based Element Generation

The objective of this phase of the algorithm is to force the generation of valid tetrahedra, if possible, even if the new elements do not satisfy the bounds used in the previous phase for element shapes. The topology-based element generation phase starts when a boundary face fails twice in trying to generate an optimal element. The list of rejected faces of the previous phase is now considered as a list of active faces and, similar to the geometry-based phase, a list of rejected faces is created for faces that eventually fail in generating valid tetrahedra.

In the topology-based element generation phase, any node close to the current base face is selected and stored in a list of candidate nodes. The node that forms the best solid angle with the base face is chosen for the generation of the new tetrahedron. If the faces of this tetrahedron do not intercept any other face of the current advancing front, the element is created and the boundary is accordingly contracted. As in the geometry-based phase, the topology-based phase ends either when the list of active faces is empty or when a face of the advancing front is rejected twice due to intersection problems.

2.2.3. Element Generation Based on Back-tracking with Face Deletion

Sometimes the procedures performed in the previous phases are not sufficient to generate a valid mesh. Consider the polyhedra shown in the centre of Fig. 8. The only possible way of generating a solid tetrahedral mesh inside this polyhedron is inserting a new node in its center. Valid elements can then be formed by connecting this node to the triangular boundary faces. There are cases, however, in which such a construction is not possible, as with the polyhedron on the left-hand side of Fig. 8. This polyhedron does not have a 'kernel region' in which any point is visible through a straight line from all its vertices [15]. In the present algorithm, the solution to this problem is to locally modify the advancing front, deleting already generated adjacent tetrahedra until a 'near' convex non-meshed polyhedron is formed. It is interesting to observe that this problem has no counterpart in two dimensions. It can be proven [16] that any non-self-intersecting polygon can be triangulated with no need to insert additional vertices.

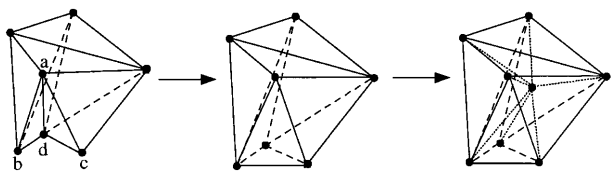

Fig. 8. Transformation of a polyhedron into a convex one.

The procedure used to transform an ill-shaped polyhedron into one with a visible kernel is as follows:

- The boundary of the ill-shaped polyhedron related to a current base face is identified.
- A visibility test is performed. This consists of computing the coordinates of the polyhedron centroid, and counting the number of intersections that would occur, for each of the polyhedron's

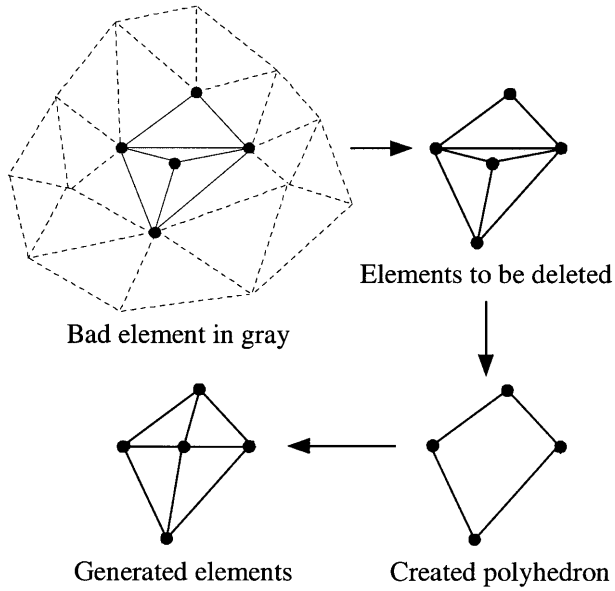

Fig. 9. Two-dimensional back-tracking procedure to remesh around a 'bad' element.

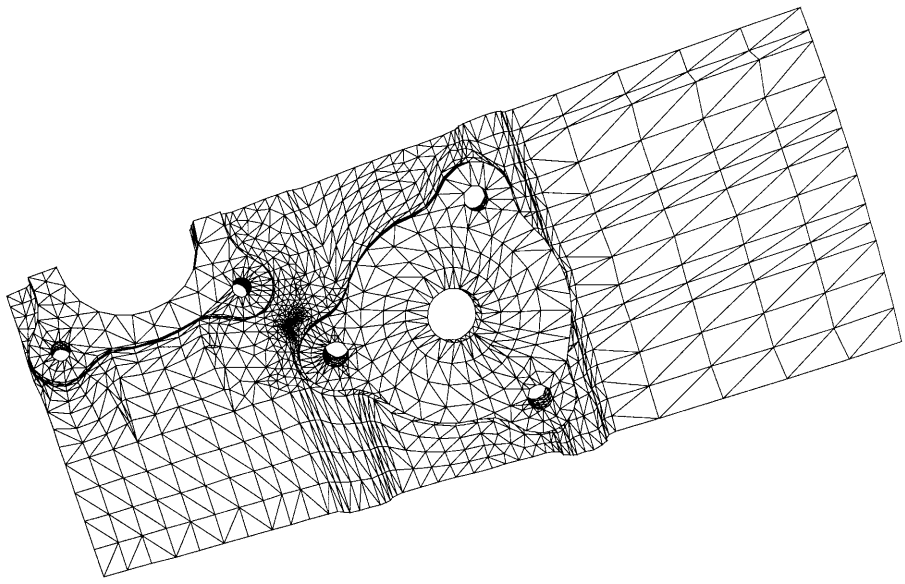

Fig. 10. Mesh for housing example.

faces, if lines were drawn from the centroid to all of the polyhedron's vertices.

- If there is at least one intersection for any of the polyhedron's faces, the polyhedron must be modified. This is done by removing the element attached to the face that has the highest number of intersections. This process is repeated until the centroid is visible from all nodes of the polyhedron.

Figure 8 illustrates the transformation of a polyhedron into a convex one after an 'intersected' face is deleted. In this example, a line drawn from node *b* to the centroid intersects face *acd*. This results in the removal of the element formed by nodes *a*, *b*, *c* and *d*. The resulting polyhedron is shown in the centre of Fig. 8. This figure also shows the tetrahedral elements generated after the insertion of a node at the centroid of the transformed polyhedron.

It is possible that the process of finding a 'near' convex polyhedron may fail if faces to be removed are part of the original boundary mesh. When this occurs, the elements attached to internal faces with non-zero intersection counts are deleted and the element extraction procedure is restarted. If this polyhedron is still not meshable, the algorithm fails and terminates. In principle, it is possible to create a boundary input mesh that forces failure of the algorithm. Such a failure, however, has not yet been observed for 'non-contrived' input, i.e. for all the

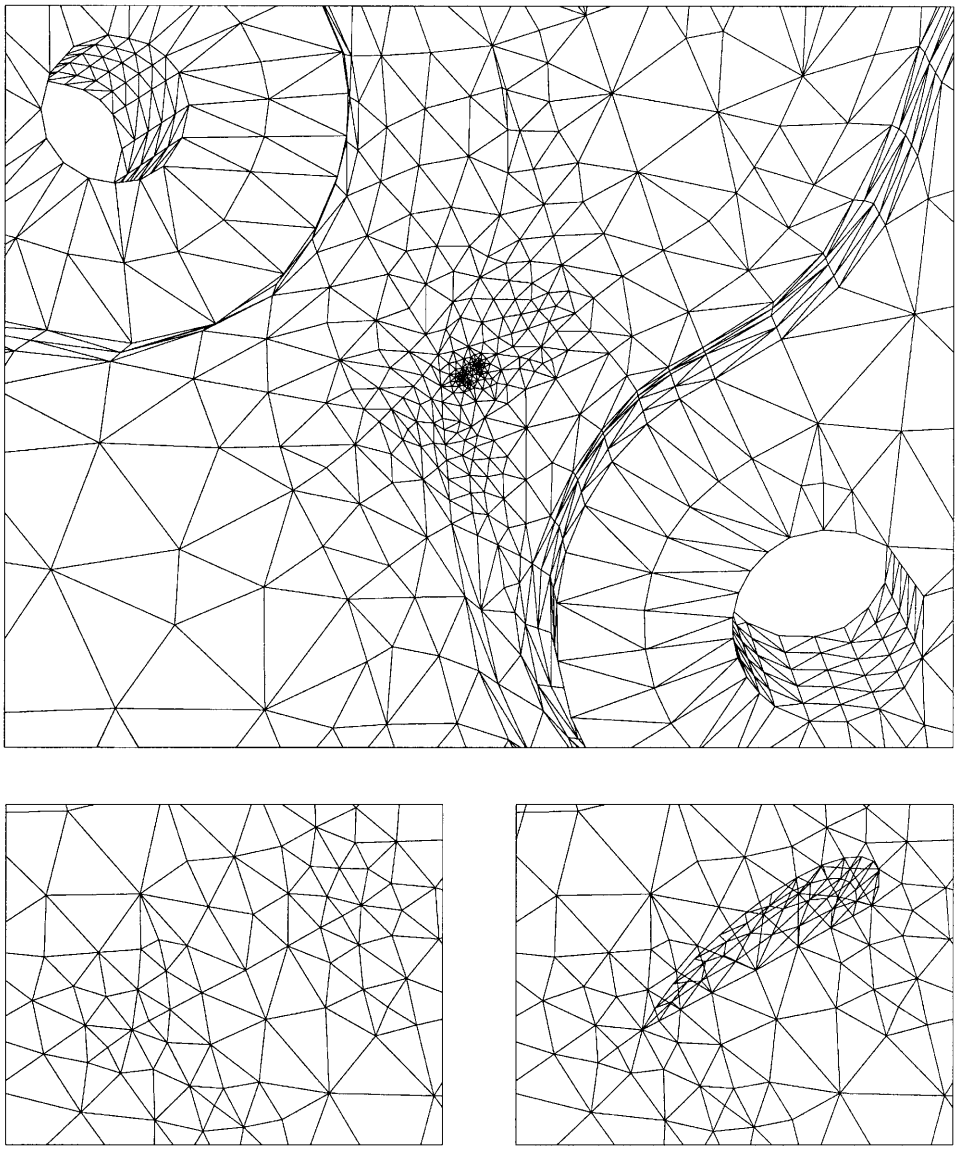

Fig. 11. Mesh detail showing an embedded crack for housing example. (a) above: detail of the crack region; (b) left: mesh of the crack region; (c) right: mesh of the crack region with the crack face mesh.

realistic input boundary meshes, such as the ones shown in Sections 3 and 4, tested so far.

2.3. Local Mesh Improvement

In the last two phases of the advancing front technique, poorly shaped tetrahedral elements might be generated. This section describes two *a posteriori* local mesh modification procedures implemented to improve mesh quality. The first is a conventional nodal relocation smoothing technique, which is based on node coordinates averaging, with validity checks. The second is a back-tracking procedure, similar to the last phase of the advancing front technique, that deletes faces of poorly shaped elements to create a region where elements with better shape can be generated.

The local mesh improvement procedures imply that element shape quality measures are necessary. This topic is well represented in the literature [17–27]. The metric adopted in this work is a normalised ratio between the root mean square of the lengths of the edges of a tetrahedron, represented by $S_{rms} = \sqrt{\frac{1}{6}\sum_{i=0}^{5} S_i^2}$, where S_i is the length of an edge, and the volume V of the tetrahedron [25]:

$$\gamma = S_{rms}^3/V \tag{1}$$

This metric generates a good quality measure and is computationally efficient. The range of valid values varies from one to infinity ([1, ∞]) and the optimal value for the regular tetrahedron is approximately 8.5.

2.3.1. Laplacian Smoothing with Validity Checks

A smoothing technique is used to improve mesh quality by relocating nodes within a patch. A general formulation for this technique is given through Eq. (2), which is a generic form of a weighted Laplacian function [28]:

$$X_O^{n+1} = X_O^n + \phi \frac{\sum_{i=1}^{m} \overline{\omega}_{iO} (X_i^n - X_O^n)}{\sum_{i=1}^{m} \overline{\omega}_{iO}} \tag{2}$$

In Eq. (2), m is the number of nodes connected to node O, X_O is the position of node O at smoothing iteration n, $\overline{\omega}_{iO}$ is the weighted function between nodes i and O, and ϕ is a relaxation parameter which is normally set in the interval [0, 1]. In theory, this generic smoothing technique provides the relaxation of interior nodes of a volume mesh and results in a mesh of better quality. In three-dimensions, however, the technique will occasionally result in invalid elements with negative volumes. It has been suggested that reducing the value of the relaxation coefficient ϕ is a way decreasing the chances that this problem will occur [27]. There are cases, however, in which this alone does not improve quality of the entire mesh. In this work, a value of $\phi = 0.5$ was adopted, and the smoothing procedures is repeated five times. At each smoothing step, after computing the new position of each internal node, a consistency test is performed. This test verifies whether any adjacent element will have a negative volume considering the new node position, in which case the relocation of this node is not performed in the current step.

2.3.2. Quality Evaluation and Local Back-tracking with Element Deletion

Laplacian smoothing is not sufficient to improve mesh quality. In this work, a back-tracking procedure is adopted to further improve the quality of a generated volume mesh. This approach can also be used for any mesh; it is not specific to the current meshing algorithm.

The back-tracking procedure consists of deleting an element that is classified as a poorly shaped element and a group of elements in its vicinity. The classification of 'bad' tetrahedra is based on a specified measure, which in this work is the γ metric,

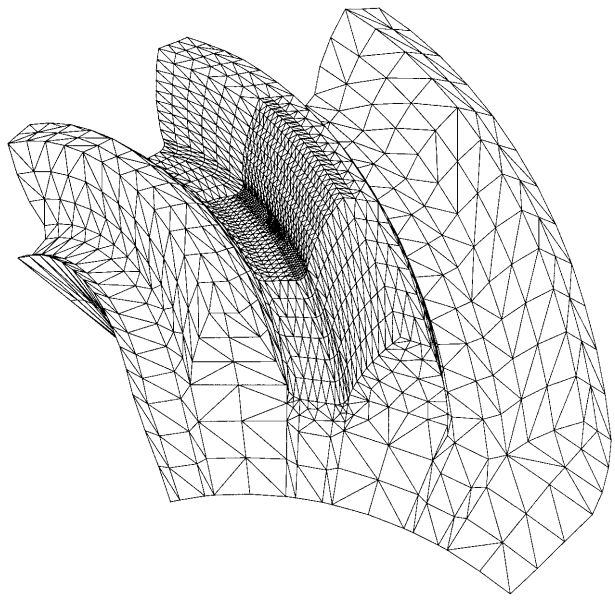

Fig. 12. Mesh for gear example.

Eq. (1). For each element of the generated mesh, the quality measure γ is evaluated. If the value of this metric is outside a pre-defined range, the element is classified as a poorly shaped element. The lower and upper bounds of this range are defined empirically, based on experiments and observations. In this work, the lower bound value is 5.0 and the upper bound is the 'optimal' metric value of 8.5 multiplied by a factor of 30.

The objective of the back-tracking procedure is to delete element faces surrounding a 'bad' element to create a local polyhedron that can be remeshed with better shaped elements. A local polyhedron to be meshed is created by deleting all elements adjacent to the 'bad' element. This is illustrated by means of Fig. 9, which shows a two-dimensional analogous case.

After the creation of the local polyhedron, an attempt is made to generate elements by inserting a new internal node in the polyhedron's centroid, as shown in Fig. 9. If this does not work, the back-tracking procedure described before is employed.

It has been observed that better results are obtained if back-tracking/regeneration is interleaved with smoothing. In the current implementation these are repeated five times.

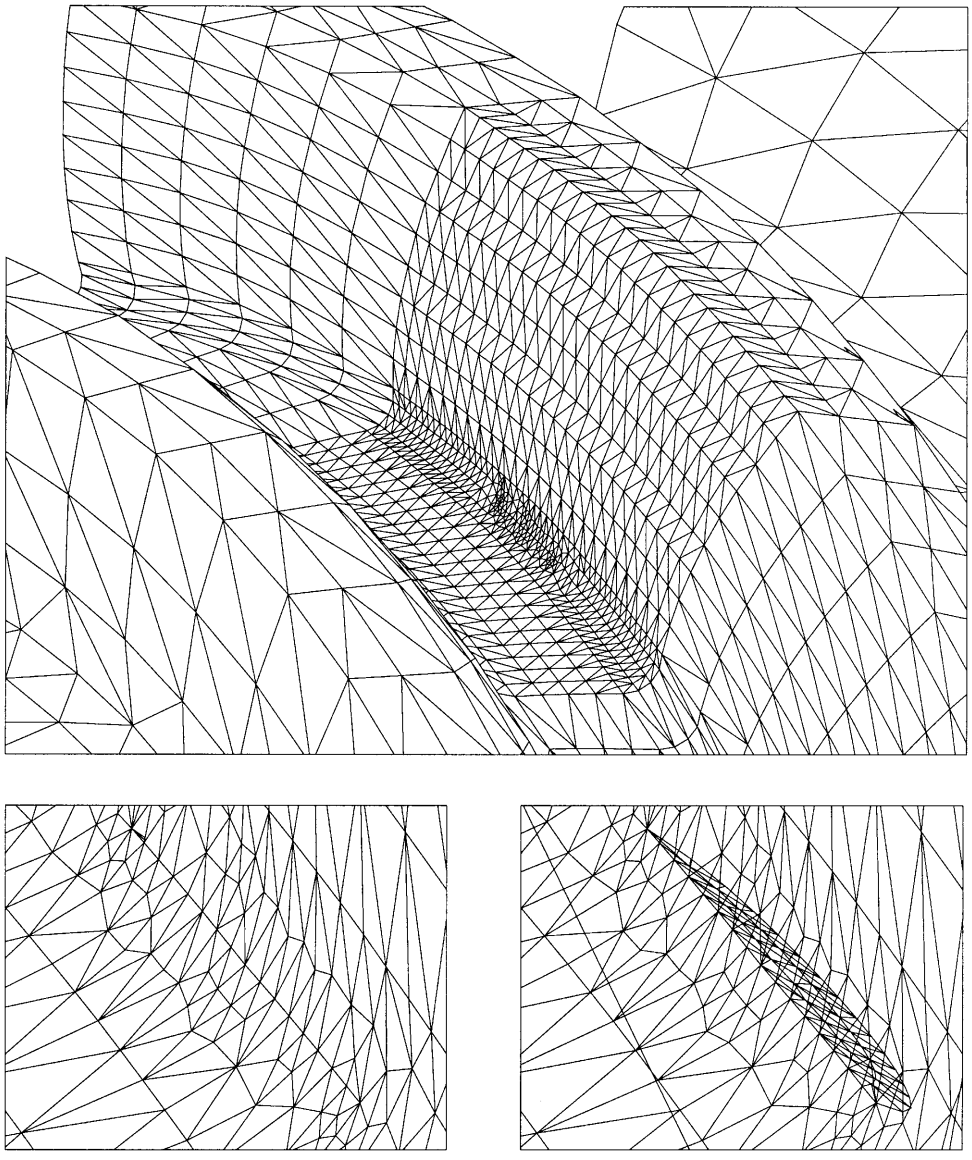

Fig. 13. Mesh detail showing an embedded crack for gear example. (a) above: detail of the crack region; (b) left: mesh of the crack region; (c) right: mesh of the crack region with the crack face mesh.

3. Mesh Quality

In this section, a study of the quality of the meshes generated by the proposed algorithm is presented. A α metric is used, which is defined by $3R_i/R_c$, where R_i and R_c are the radii of the inscribed and circumscribed spheres, respectively. This metric is equal to 1.0 for an optimal element, and sliver elements have values lower than 0.1. The α metric, instead of the metric γ described in the previous section, and used within the algorithm, was adopted in the present study because its interpretation is more intuitive, and it is more widely referenced in the literature.

In this study, three models are considered: a portion of a housing, a portion of a spiral-bevel gear, and a portion of a turbofan hub (Figs 10–15). All three examples contain small surface cracks.

Histograms of the number of the elements generated in various ranges of α are shown in Figs 16–18. These figures show the distributions both before (left bar) and after (right bar) the application of the quality improvement procedures (back-tracking and smoothing). Table 1 presents some statistics related to α for the three examples.

It can be seen that the majority of the elements are all located in the range [0.7, 0.8], which represents well-shaped elements, even before the application of the quality improvement procedures. However, there are a significant number of elements (1.76–3.77%) with α values lower than 0.1, which represent undesirable elements. After the application of the quality improvement procedures, the number of poorly-shaped elements drops significantly (0.49–0.90%). It is important to mention that most of these poorly-shaped elements are located in the interior of the model, away from the crack front or stress concentration regions, where better elements are desirable. This happens because as the algorithm advances from the boundary, it is very likely that the better elements will be generated near it, where the geometry-based phase can be employed.

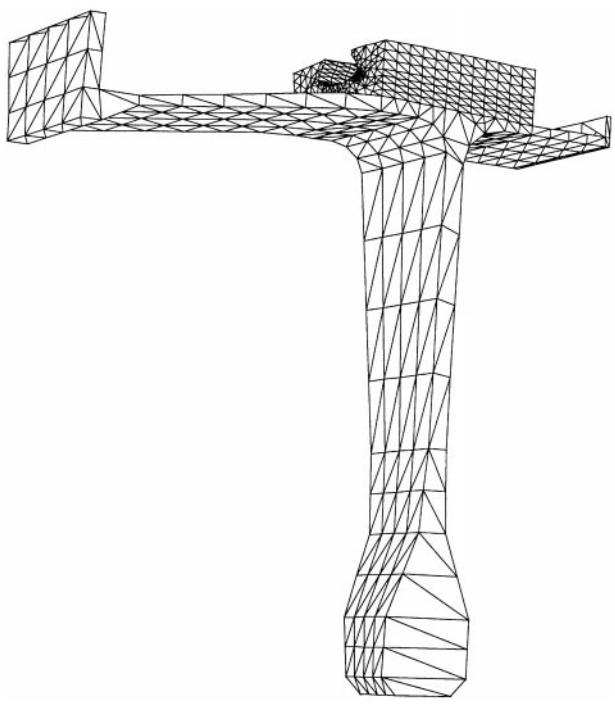

Fig. 14. Mesh for turbofan example.

4. Mesh Generation Performance

A formal analysis of the computational complexity of the proposed algorithm would be very difficult, especially considering input data specific steps such as back-tracking. Nevertheless, a realistic estimate of the expected performance of the algorithm is very important for its practical use. In this section an *informal* empirical study of the algorithm performance is presented.

A computational complexity of $O(NlogN)$ has been reported in the literature for advancing front techniques, N being the number of generated elements [7,29]. It has also been reported that a performance of $O(N^p logN)$, with p slightly larger than one, is a more realistic measure [8,9]. This computational complexity is used here.

For the present study, six discretisations for two geometries were considered. The first example is a torus, where the first and the final meshes are shown in Fig. 19. A similar problem was also analysed by other work [8], and a $CN^p logN$ performance was reported. The second example is a cube with an inclined interior cylindrical void. The first and final meshes are also shown in Fig. 21. Performance was measured as the wallclock time for the program running on an IBM 43P workstation, Model 260.

A least square fit of the measured times to the equation $time = CN^p logN$ was performed for the two geometries. The resulting fitting parameters are $C = 0.0000629$ and $p = 1.21$ for the torus, and $C = 0.000269$ and $p = 1.25$ for the cube. These are plotted in Figs 20 and 22.

It can be seen from the plots of Figs 20 and 22 that, in both cases, these equations model the performance of the algorithm well over the range of mesh sizes studied. However, the parameters computed for the two models differ greatly, especially for the values of C. The rate of generation

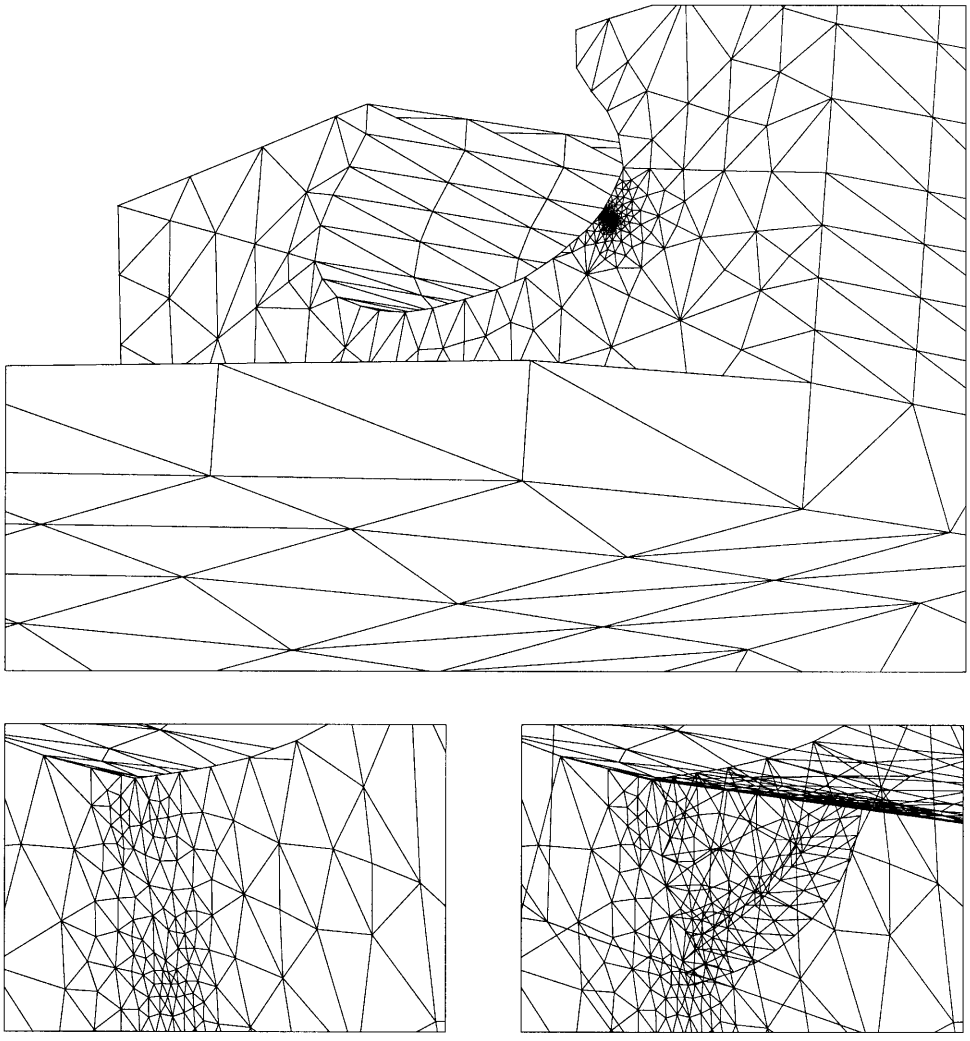

Fig. 15. Mesh detail showing an embedded crack for turbofan example. (a) above: detail of the crack region; (b) left: mesh of the crack region; (c) right: mesh of the crack region with the crack face mesh.

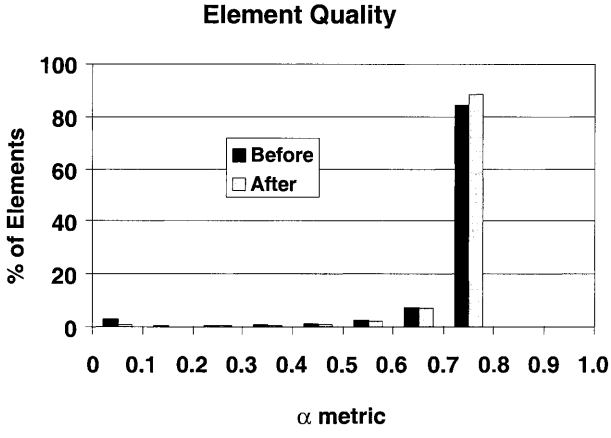

Fig. 16. Histogram of element quality for housing example.

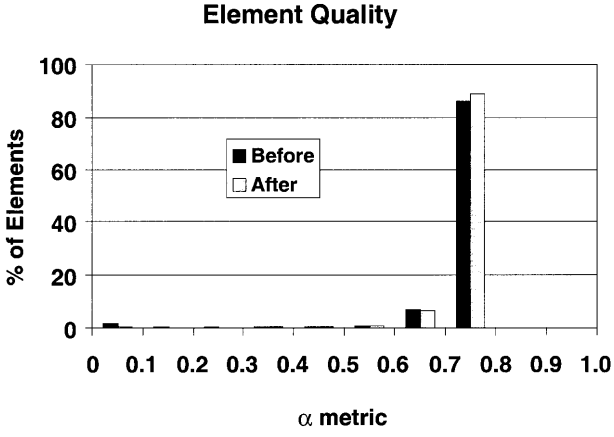

Fig. 17. Histogram of element quality for gear example.

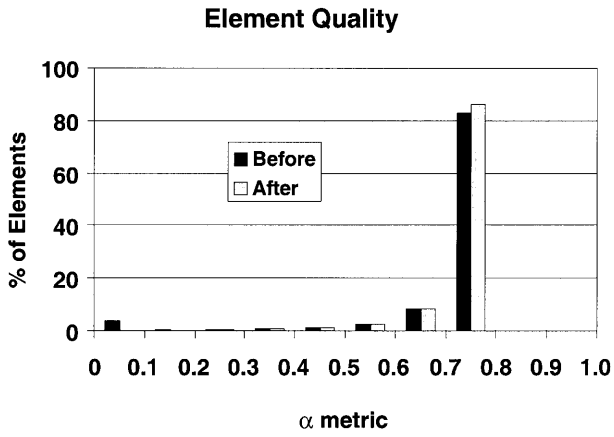

Fig. 18. Histogram of element quality for turbofan example.

for the torus is on the order of 150–200 tetrahedra per second, which is acceptable, while the rate of the cube is on the order of 20–80 tetrahedra per second, which is a relatively 'low rate'. This highlights the influence of the input data in the performance of the algorithm. As expected, longer runtimes are required for more complex input geometries. Nevertheless, this 'low' rate indicates that it would be very useful to use some auxiliary data structure for search operations, such as trees, to optimise the element generation for some models.

5. Conclusions

An algorithm for generating unstructured tetrahedral meshes for arbitrarily shaped three-dimensional regions was described. The algorithm incorporates aspects of well known meshing procedures, and includes some original steps. The algorithm works for regions without cracks as well as for regions with one or multiple cracks. The algorithm was designed to meet four specific requirements:

- It should avoid producing elements with poor aspect ratios.
- It can generate meshes which conform to existing triangular meshes on the boundary of a domain.
- It generates meshes which exhibit good transitions between regions of different element sizes.
- It works properly for cases where distinct boundary nodes are geometrically coincident (e.g. nodes on opposite faces of a crack).

The input to the algorithm is a triangular surface mesh, which describes the domain to be meshed. The steps in the algorithm are as follows:

- An octree is generated to control the distribution of node points generated in the interior.
- A two-pass advancing front procedure is used to generate elements. On the first pass, elements are generated based on geometrical criteria, which produce well shaped elements. On the second pass, elements are generated based only on the criterion that they have valid topology.
- If the advancing front procedure cannot proceed, a back-tracking strategy is employed where some elements are deleted, and the advancing front is restarted.
- Once a valid mesh is generated, the quality of the element shapes is improved by the use of standard Laplacian smoothing, and by locally deleting poorly shaped elements and those adjacent to them, and then restarting the boundary contraction.

A number of realistic examples were presented to demonstrate the distribution in the quality of the generated elements. It was shown that only a small percentage (0.49–0.90%) of the total generated elements are poorly shaped. These poorly shaped elements are usually located in the interior of the model, away from the crack front and stress concentration regions where better elements are desired.

Two examples were presented to determine empirical measures of the algorithm performance. It was shown that the algorithm has the expected performance for practical applications, although

Table 1. Statistical values related to the quality for all examples

Example	Histogram	#Elements	α_{avg}	α_{min}	α^{max}
Housing	Before	16463	0.675	0.025	0.729
Housing	After	17043	0.696	0.023	0.740
Gear	Before	17386	0.684	0.025	0.738
Gear	After	16990	0.699	0.033	0.742
Turbofan	Before	9628	0.668	0.018	0.733
Turbofan	After	10046	0.692	0.022	0.741

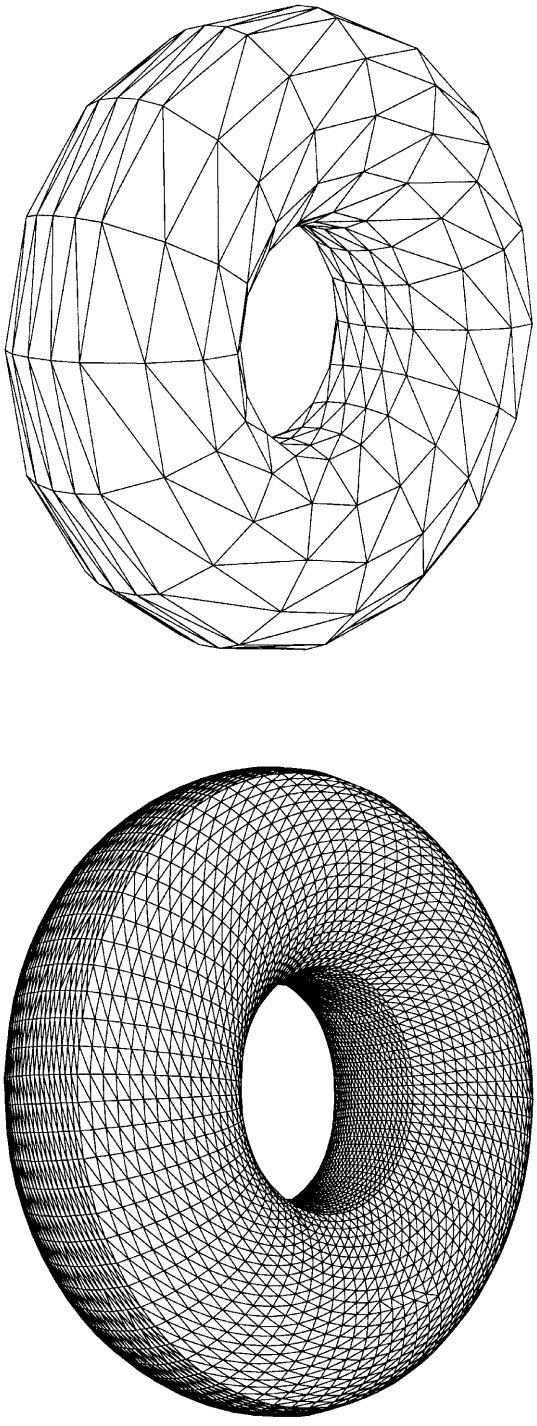

Fig. 19. Initial and final discretisations of a torus.

auxiliary search structures, such as trees, could be used in order to improve the algorithm's performance.

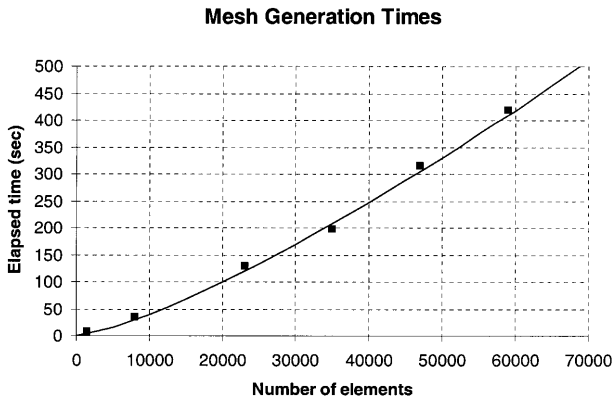

Fig. 20. Generation times for the torus.

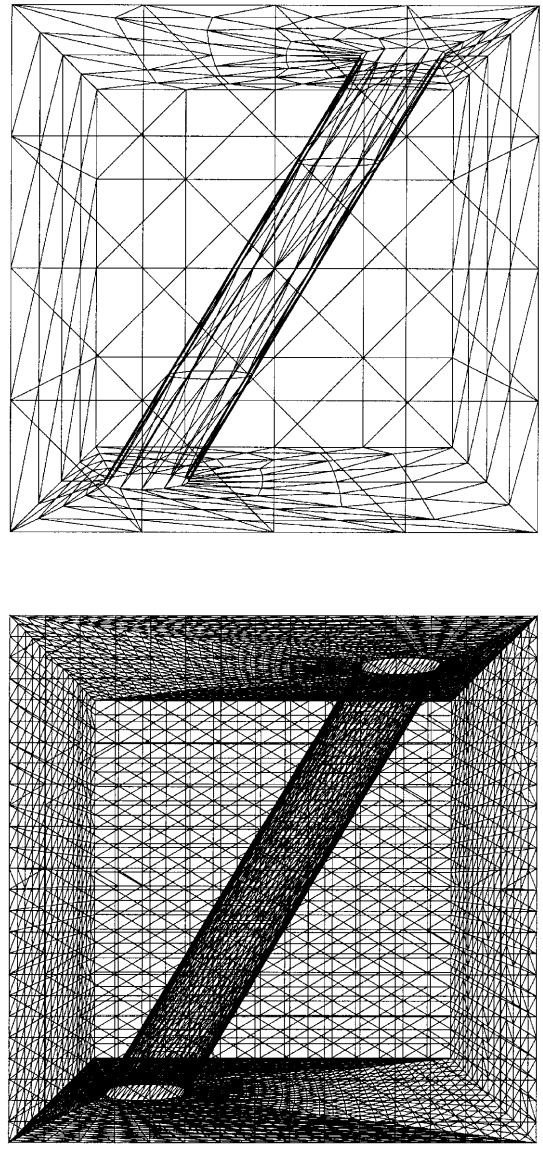

Fig. 21. Initial and final discretisations of a cube with a cylindrical void.

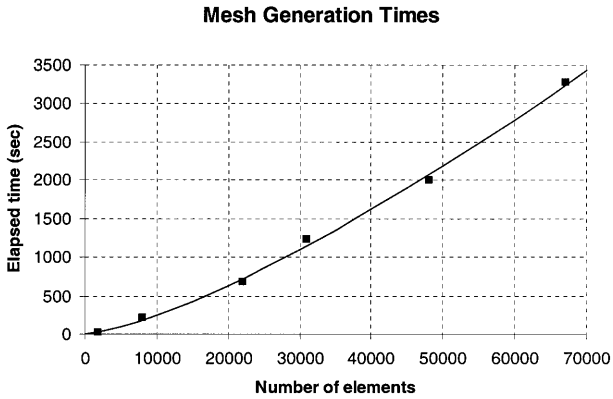

Fig. 22. Generation times for the cube with a cylindrical void.

Acknowledgements

The first and the third authors acknowledge fellowships provided by Brazilian agencies CNPq and CAPES, respectively. The fourth author acknowledges the financial support from Brazilian agency Finep through a RECOPE project. The second and fifth authors acknowledge the support from the National Science Foundation for project NSF CCISE Award 9726388 – Crack Propagation on Teraflop Computers – and the discussions with Prof. Steve Vavasis and Dr. Paul Chew.

References

1. Wawrzynek, P. A., Ingraffea, A. R. (1987) Interactive finite element analysis of fracture processes: an integrated approach. Theoretical and Applied Fracture Mechanics, 8, 137–150
2. Lau, T. S., Lo, S. H. (1996) Finite element mesh generation over analytical curved surfaces. Computers & Structures, 59, 301–309
3. Lewis, R. H., Zheng, Y., Gethin, D. T. (1996) Three-dimensional unstructured mesh generation: Part 2. Surface meshes. Computer Methods in Applied Mechanics, 134, 269–284
4. Gomes Coelho, L. C., Gattass, M., Figueiredo, L. H. (2000) Intersecting and trimming parametric meshes on finite element shells. International Journal for Numerical Methods in Engineering, 47, 777–800
5. Anastasiou, K., Chan, C. T. (1996) Automatic triangular mesh generation scheme for curved surfaces. Communications in Numerical Methods in Engineering, 12, 197–208
6. Peraire, J., Peiro, J., Formaggia, L., Morgan, K., Zienkiewicz, O. C. (1988) Finite Euler computation in three-dimensions. International Journal for Numerical Methods in Engineering, 26, 2135–2159
7. Lohner, R., Parikh, P. (1988) Generation of three-dimensional unstructured grids by the advancing-front method. International Journal for Numerical Methods in Fluids, 8, 1135–1149
8. Jin, H., Tanner, R. I. (1993) Generation of unstructured tetrahedral meshes by advancing front technique. International Journal for Numerical Methods in Engineering, 36, 1805–1823
9. Moller, P., Hansbo, P. (1995) On advancing front mesh generation in three dimensions. International Journal for Numerical Methods in Engineering, 38, 3551–3569
10. Chan, C. T., Anastasiou, K. (1997) An automatic tetrahedral mesh generation scheme by the advancing front method. Communications in Numerical Methods in Engineering, 13, 33–46
11. Rassineux, A. (1998) Generation and optimization of tetrahedral meshes by advancing front technique. International Journal for Numerical Methods in Engineering, 41, 651–674
12. Carter, B. J., Wawrzynek, P. A., Ingraffea, A. R. (2000) Automated 3-D crack growth simulation. International Journal for Numerical Methods in Engineering, 47, 229–253
13. Shephard, M. S., Georges, M. K. (1991) Automatic tri-dimensional mesh generation by the finite octree technique. International Journal for Numerical Methods in Engineering, 32, 709–749
14. Carvalho, P. C. P., Cavalcanti, P. R. (1995) Point in polyhedron testing using spherical polygons. In: Paeth, A. W. (Editor), Graphics Gems V. Academic Press, 709–749
15. Perucchio, R., Saxena, M., Kela, A. (1989) Automatic mesh generation from solid models based on recursive spatial decomposition. International Journal for Numerical Methods in Engineering, 28, 2469–2501
16. O'Rourke, J. (1987) Art Gallery Theorems and Algorithms. Oxford University Press
17. Bramble, J. H., Zlamal, M. (1970) Triangular elements in the finite element method. The Mathematics of Computations, 24, 809–810
18. Ciarlet, P. G. (1970) Orders of convergence in finite element method. The Mathematics of Finite Elements and Applications, 1, 59–82
19. Babuska, I., Aziz, A. K. (1976) On the angle condition in the finite element method. SIAM Journal of Numerical Analysis, 13, 214–226
20. Cavendish, J. C., Field, D. A., Frey, W. H. (1995) An approach to automatic three-dimensional finite element mesh generation. International Journal for Numerical Methods in Engineering, 21, 329–347
21. Cougny, H. L., Shephard, M. S., Georges, M. K. (1990) Explicit node smoothing within octree, Troy-NY, Report 10–1990, SCOREC, Rensselaer Polytechnic Institute
22. Dannelongue, H. H., Tanguy, P. A. (1991) Three-dimensional adaptive finite element computations and applications to non-Newtonian flows. International Journal for Numerical Methods in Engineering, 13, 145–165
23. Joe, B. (1991) Delaunay versus Max-Min solid angle triangulations for three-dimensional mesh generation. International Journal for Numerical Methods in Engineering, 31, 987–997
24. Parthasarathy, V. N., Graichen, C. M., Hathaway, A. F. (1993) A comparison of tetrahedron quality measures. Finite Element Analysis and Design, 15, 255–261
25. Weatherill, N. P., Hassan, O. (1994) Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. Inter-

national Journal for Numerical Methods in Engineering, 37, 2005–2039
26. Liu, A., Joe, B. (1994) Relationship between tetrahedron shape measures. BIT, 34, 268–287
27. Lewis, R. H., Zheng, Y., Gethin, D. T. (1996) Three-dimensional unstructured mesh generation: Part 3. Volume meshes. Computer Methods in Applied Mechanics, 134, 285–310
28. Foley, J. D., van Dam, A., Feiner, S. K., Hughes, J. F. (1990) Computer Graphics (Principles and Practice). Addison-Wesley, Reading, MA
29. Bonet, J., Peraire, J. (1991) An Alternativing Digital Tree (ADT) algorithm for 3D geometric search and intersection problems. International Journal for Numerical Methods in Engineering, 31, 1–17