

## Blossom-Quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm

J.-F. Remacle<sup>1,\*</sup>, J. Lambrechts<sup>1</sup>, B. Seny<sup>1</sup>, E. Marchandise<sup>1</sup>, A. Johnen<sup>2</sup> and C. Geuzainet<sup>2</sup>

<sup>1</sup>*Institute of Mechanics, Materials and Civil Engineering (iMMC), Université Catholique de Louvain, Bâtiment Euler, Avenue Georges Lemaître 4, 1348 Louvain-la-Neuve, Belgium*

<sup>2</sup>*Department of Electrical Engineering and Computer Science, Université de Liège, Montefiore Institute B28, Grande Traverse 10, 4000 Liège, Belgium*

### SUMMARY

A new indirect way of producing all-quad meshes is presented. The method takes advantage of a well-known algorithm of the graph theory, namely the Blossom algorithm, that computes the minimum-cost perfect matching in a graph in polynomial time. The new Blossom-Quad algorithm is compared with standard indirect procedures. Meshes produced by the new approach are better both in terms of element shape and in terms of size field efficiency. Copyright © 2011 John Wiley & Sons, Ltd.

Received 15 December 2010; Revised 13 July 2011; Accepted 13 July 2011

KEY WORDS: quadrilateral meshing; surface remeshing; graph theory; optimization; perfect matching

### 1. INTRODUCTION

Quadrilateral surface meshes are sometimes considered as superior to triangular meshes for finite element simulations. Discussions about if and why quadrilaterals are better than triangles are usually passionate in the finite element community. We will not try to argue about that thorny question here—but we assume that quadrilateral meshes are indeed useful, and in this paper, we present a new way of generating such meshes.

Let us first briefly recall which kinds of methods can be used to build non-uniform quadrilateral meshes in an automatic manner. There are essentially two categories of methods.

In *direct methods*, the quadrilaterals are constructed at once, either using some kind of advancing front technique [1] or using regular grid-based methods (quadtrees). Advancing front methods for quads are considered to be non-robust, and quadtree methods usually produce low-quality elements close to the boundaries of the domain and are unable to fulfill general size constraints (anisotropy, strong variations).

In *indirect methods*, a triangular mesh is built first. Triangle-merge methods then use the triangles of the initial mesh and recombine them to form quadrangles [2, 3]. Other more sophisticated indirect methods use a mix of advancing front and triangle merge [4].

The method we present here is an indirect approach to quadrilateralization. We make use of a famous algorithm of the theory of graphs: the Blossom algorithm, proposed by Edmonds in 1965 [5, 6], which allows us to find the minimum-cost perfect matching of a given graph. The new method has some clear advantages: (i) it provides a mesh that is guaranteed to be quadrilateral only; (ii) it is optimal in a certain way; and (iii) it is fast.

\*Correspondence to: J.-F. Remacle, Université Catholique de Louvain, Bâtiment Euler, Avenue Georges Lemaître 4, 1348 Louvain-la-Neuve, Belgium.

†E-mail: jean-francois.remacle@uclouvain.be

## 2. MESH QUALITY MEASURES

The aim of the mesh generation process described in this paper is to build a mesh made of quadrilaterals that has controlled element sizes and shapes. We are interested in generating non-uniform quadrilateral meshes. Local information about sizes is given through the definition of a mesh size field that returns, for every point  $\vec{x}$  in the domain, a ‘characteristic’ length  $h(\vec{x})$  that has to be fulfilled by the mesh.

Let  $\vec{a}$  and  $\vec{b}$  be two points of  $\mathbb{R}^3$ . The adimensional length of the vector  $\vec{ab}$  with respect to the non-uniform size field  $h$  is defined as follows [3, 7]:

$$l^h(\vec{ab}) = \|\vec{ab}\| \int_0^1 \frac{1}{h(\vec{a} + t\vec{ab})} dt. \quad (1)$$

An optimum mesh in terms of the size is a mesh for which every edge  $i$  is of adimensional size  $l_i^h$  equal to 1. It is of course impossible to have such a perfect unit mesh. Here, we define the *efficiency index* [7]  $\tau$  of a mesh as the exponential of the mean value of the difference between each edge length  $l_i^h$  and 1:

$$\tau[\%] = 100 \exp\left(\frac{1}{n_e} \sum_{i=1}^{n_e} d_i\right), \quad (2)$$

with  $d_i = l_i^h - 1$  if  $l_i^h < 1$ ,  $d_i = (1/l_i^h) - 1$  if  $l_i^h > 1$ , and  $n_e$  the number of edges in the mesh. Surface mesh algorithms usually produce triangular meshes with typical values of  $\tau$  around 85%, that is, with non-dimensional sizes around  $1/\sqrt{2} \leq l_i^h \leq \sqrt{2}$ .

Having the right sizing for the mesh is not enough: mesh generators should also provide meshes with controlled element qualities. We then define a quality measure for quadrilateral elements. Consider a quadrilateral element  $q$  and its four internal angles  $\alpha_k$ ,  $k = 1, 2, 3, 4$ . We define the quality  $\eta(q)$  of  $q$  as follows:

$$\eta(q) = \max\left(1 - \frac{2}{\pi} \max_k \left(\left|\frac{\pi}{2} - \alpha_k\right|\right), 0\right). \quad (3)$$

This quality measure is 1 if the element is a perfect quadrilateral and is 0 if one of those angles is either  $\leq 0$  or  $\geq \pi$ . In what follows, we will present statistics for the quadrilateral meshes:

- The efficiency index  $\tau$ , which measures the adequacy of the mesh with the mesh size field. The index  $\tau$  is smaller or equal to 1 and should be as close as possible to  $\tau = 100\%$ .
- The average element quality  $\bar{\eta}$  as well as the worst element quality  $\eta_w$ , which can be important in the context of finite element simulations.

## 3. INDIRECT QUADRILATERALIZATION USING A NON-OPTIMAL MATCHING ALGORITHM

In Section 1, we have made the distinction between direct and indirect methods for the construction of quadrilateral meshes. In the case of indirect methods, a triangular mesh is first constructed. Then, triangles are recombined in order to produce quadrangles.

Consider a triangular mesh made of  $n_t$  triangles  $t_i$ ,  $i = 1, \dots, n_t$ . In what follows, we consider internal edges  $e_{ij}$  of the mesh that are common to triangles  $t_i$  and  $t_j$ . We define a cost function  $c(e_{ij}) = 1 - \eta(q_{ij})$  that is associated to each graph edge  $e_{ij}$  of the mesh and that is defined as the mesh quality of the quadrilateral  $q_{ij}$  that is formed by merging the two adjacent triangles  $t_i$  and  $t_j$ . Usual indirect quadrilateralization procedures work as follows [3]. Edges  $e_{ij}$  of the graph are sorted with respect to their individual cost functions. Then, the two triangles that are adjacent to the best edge  $e_{ij}$  of the list are recombined into a quadrilateral. Triangles  $t_i$  and  $t_j$  are tagged in order to prevent other edges that are adjacent either to  $t_i$  or to  $t_j$  from being used for another

quadrilateral forming. Then, the algorithm processes the ordered list of edges, forming quadrilaterals with triangles adjacent to an edge as long as none of those adjacent triangles are tagged. Figure 1 shows an illustration of this procedure for a rectangular domain of size  $1 \times 3$  and a mesh size field defined by

$$h(x, y) = 0.1 + 0.08 \sin(3x) \cos(6y).$$

Isolated triangles inevitably remain in the mesh, and the resulting mesh is not made of quadrilaterals only. The mesh is then said to be *quad dominant*. In the example of Figure 1, the resulting mesh is made of 836 quads and 240 triangles.

A mesh composed of quadrilaterals can be built subsequently using a uniform mesh refinement procedure [3]. Every quadrilateral of the quad-dominant mesh is split into four sub-quadrilaterals, and every triangle is split into three sub-quadrilaterals (Figure 2). For the size criterion  $h(\vec{x})$  to be fulfilled, the initial triangular mesh should thus be built using a size field with twice the value (i.e.,  $2h$ ) that is expected in the final mesh.

The recombination process just described is sub-optimal. It does not provide the best set of edges to be recombined with respect to some general cost function. Indeed, the only optimality property of this algorithm is that it ensures that the best triangle pair will be recombined.

The second part of the algorithm, namely the mesh refinement step, also has some drawbacks. Splitting every element of the mesh produces a mesh that has half the size of the initial mesh. It is of course possible to generate an initial mesh with double the required size. Yet, with real geometries, the new vertices will have to be added on the geometry, which is not trivial. On the other hand, the refinement step does not allow a sharp control of the mesh size. On Figure 2, the procedure ends with an efficiency index of 79%, which cannot be considered as good.

In [2], the authors proposed a scheme for recombining triangular meshes that does not always require the refinement step, using a kind of advancing front technique. The merging of triangles starts at the boundary; when a front closes, the algorithm attempts to maintain an even number of triangles on any sub-front. Again, this approach is sub-optimal because the result depends on the ordering of elements and on the choice of the initial front.

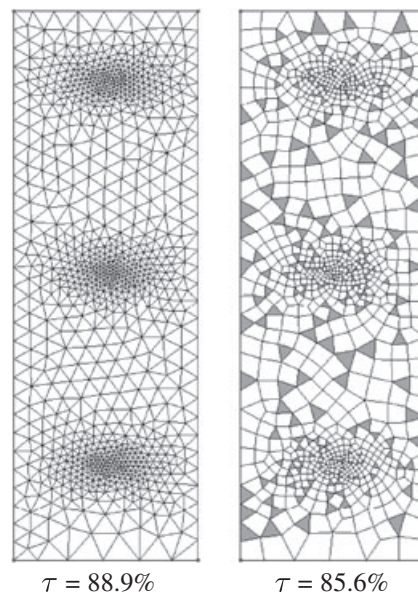


Figure 1. Illustration of the quad-dominant algorithm. The left mesh is the initial triangular mesh, and the right mesh is the quad-dominant mesh, after smoothing (triangles are in gray).

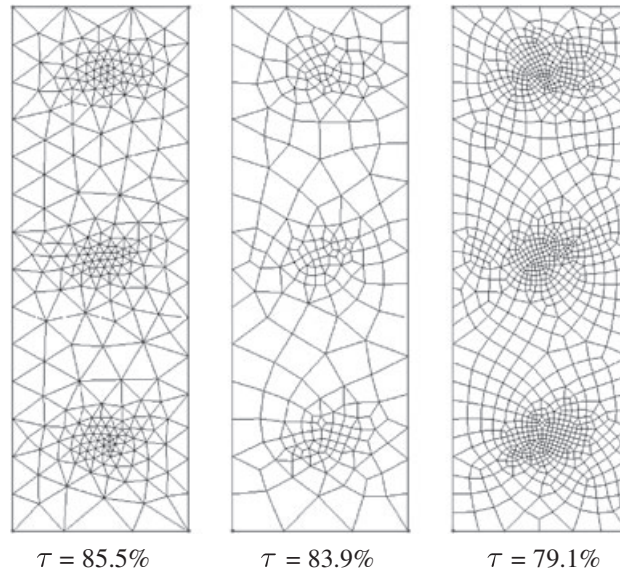


Figure 2. A quad-dominant algorithm using  $h^{\text{algo}} = 2h$  followed by a one-mesh refinement procedure leads to a reduction of the efficiency index  $\tau$ .

#### 4. THE NEW BLOSSOM-QUAD ALGORITHM

Here, our aim is to build a mesh generation scheme that starts with a triangular mesh and attempts to find the set of pairs of triangles that forms the best possible quadrilaterals with the constraint of not leaving any remaining triangle in the mesh.

##### 4.1. Blossom: a minimum-cost perfect-matching algorithm

Let us consider  $G(V, E, c)$  as an undirected weighted graph. Here,  $V$  is the set of  $n_V$  vertices,  $E$  is the set of  $n_E$  undirected edges, and  $c(E) = \sum c(e_{ij})$  is an edge-based cost function, that is, the sum of all weights associated to every edge  $e_{ij} \in E$  of the graph. A *matching* is a subset  $E' \subseteq E$ , such that each node of  $V$  has at most one incident edge in  $E'$ . A matching is said to be perfect if each node of  $V$  has exactly one incident edge in  $E'$ . As a consequence, a perfect matching contains exactly  $n_{E'} = n_V/2$  edges. A perfect matching can therefore only be found for graphs with an even number of vertices. A matching is optimum if  $c(E')$  is minimum among all possible perfect matchings.

In 1965, Edmonds [5, 8] invented the *Blossom algorithm* that solves the problem of optimum perfect matching in polynomial time. A straightforward implementation of Edmonds' algorithm requires  $\mathcal{O}(n_V^2 n_E)$  operations.

Since then, the worst-case complexity of the Blossom algorithm has been steadily improving. Both Lawler [9] and Gabow [10] achieved a running time of  $\mathcal{O}(n_V^3)$ . Galil *et al.* [11] improved it to  $\mathcal{O}(n_V n_E \log(n_V))$ . The current best-known result in terms of  $n_V$  and  $n_E$  is  $\mathcal{O}(n_V(n_E + \log n_V))$  [12].

There is also a long history of computer implementations of the Blossom algorithm, starting with the Blossom I code of Edmonds *et al.* [6]. In this paper, our implementation makes use of the Blossom IV code of Cook and Rohe [13],<sup>‡</sup> which has been considered for several years as the fastest available implementation of the Blossom algorithm.

<sup>‡</sup>Computer code available at <http://www2.isye.gatech.edu/~wcook/blossom4/>

4.2. Optimal triangle merging

Consider now a mesh made of  $n_t$  triangles and  $n_v$  vertices. Consider a specific weighted graph  $G(V, E, c)$  that is built using triangle adjacencies in the mesh. Here, every vertex of the graph is a triangle  $t_i$  of the mesh, and every edge of the graph is an internal edge  $e_{ij}$  of the mesh that connects two neighboring triangles  $t_i$  and  $t_j$ . Figure 3 shows a simple triangular mesh with its graph and one perfect matching.

Let us come back first to the non-optimal triangle-merging algorithms of Section 3. In terms of what has just been defined, the subset  $E'$  of edges that have been used for triangle merging in the approach of Borouchaki and Frey [3] is a matching that is very rarely perfect. The one of Lee and Lo [2] is usually a perfect matching, but not necessarily the optimal one.

Here, we propose a new indirect approach to quadrilateral meshing that takes advantage of the Blossom algorithm of Edmonds. To this end, we apply the Blossom IV algorithm to the graph of the mesh. We intend to find the optimum perfect matching with respect to the following total cost function:

$$c = \sum_{e \in E'} (1 - \eta(q_{ij})), \tag{4}$$

that is, the sum of all elementary cost functions (or ‘badnesses’) of the quadrilaterals, which results in the merging of the edges of the perfect matching  $E'$ .

An obvious requirement for the final mesh to be quadrilateral only is that the initial triangular mesh contains an even number of triangles (i.e., an even number of graph vertices). Euler’s formula for planar triangulations states that the number of triangles in the mesh is

$$n_t = 2(n_v - 1) - n_v^b, \tag{5}$$

where  $n_v^b$  is the number of mesh nodes on its boundary. So, the number of mesh points on the boundary  $n_v^b$  should be even. Here, our algorithms are applied to general solid models that have a boundary representation [14]. This means that model surfaces are bounded by connected model

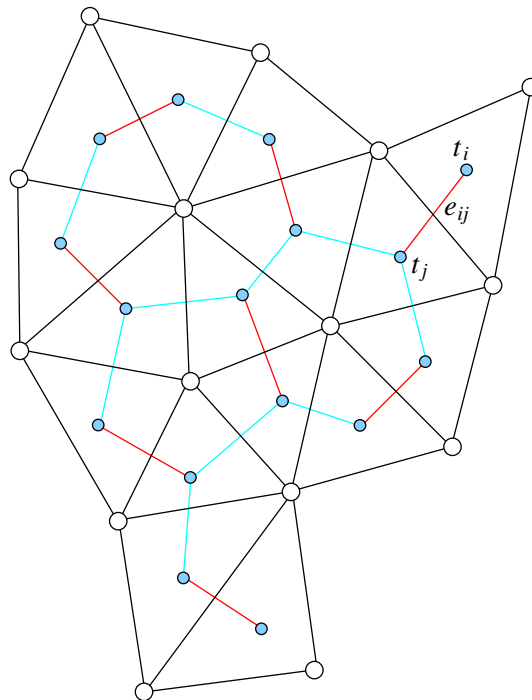


Figure 3. A mesh (in black) and its graph (in cyan and red). The set of graph edges in red forms a perfect matching.

edges that form edge loops and that the model edges are bounded by model vertices. The mesh vertices of a model edge  $n_v^{b_i}$  are defined as the mesh vertices on that edge minus the model vertices. The total number of mesh points on the boundary  $n_v^b$  can thus be written as follows:

$$n_v^b = \sum_{i=0}^{N_E} (1 + n_v^{b_i}). \tag{6}$$

It is then easy to see that for  $n_v^b$  to be even, it is sufficient for  $n_v^{b_i}$  to be odd. This means that a sufficient condition for having an even number of triangles in the mesh is to have every model edge  $b_i$  discretized with an odd number of mesh vertices.

Figure 4 shows the same illustrative example as Figures 1 and 2 using the Blossom algorithm for recombining the triangles together with the optimization procedure that will be described in Section 6. The final result is much better not only with respect to the efficiency index (with  $\tau = 83\%$ ) but also with respect to the worst element quality ( $\eta_w = 0.405$  instead of  $\eta_w = 0.310$  for the mesh of Figure 2). The average quality is better as well.

#### 4.3. Alternative cost functions

Quadrilateral meshes are not isotropic by definition: this is an important difference between quadrilateral and triangular meshes. In principle, the orientation of the mesh can be defined through a ‘cross field’, that is, a field of orthogonal tangent vectors to the surface. The cross field can be used to orient the edges of the quadrilateralization [15].

It is possible to define an alternative cost function  $c$  that is based on the information contained in the cross field. Consider a graph edge  $e_{ij}$  that is tangent to the surface with a unit vector  $\vec{e}_{ij}$ . Assume that the cross field at the midpoint of  $e_{ij}$  is defined through two orthogonal unit tangents  $\vec{t}_1$  and  $\vec{t}_2$ . Edges that are aligned with one of the directions of the cross field should not be used for triangle merging. This leads to the following alternative cost function:

$$c(e_{ij}) = \frac{1}{1 - \sqrt{2}/2} \left( \max(|\vec{e}_{ij} \cdot \vec{t}_1|, |\vec{e}_{ij} \cdot \vec{t}_2|) - \sqrt{2}/2 \right). \tag{7}$$

This edge cost function is maximum ( $c(e_{ij}) = 1$ ) for graph edges aligned with one of the directions of the cross field and is minimum ( $c(e_{ij}) = 0$ ) for graph edges that are aligned with the bisector of

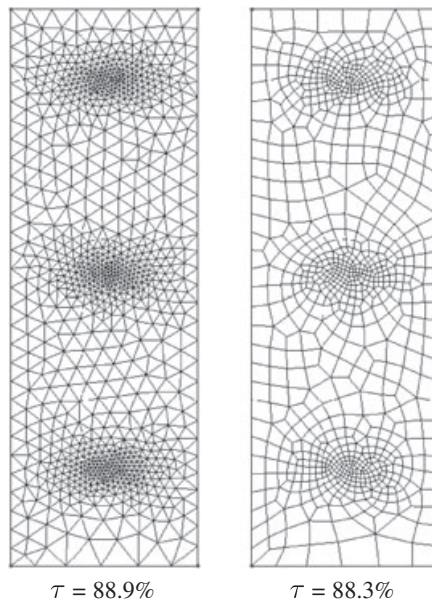


Figure 4. Illustration of the Blossom-Quad algorithm.

the cross field. For this minimal value of edge cost function, the mesh edges will then be aligned with the orthogonal unit tangents  $\vec{t}_1$  and  $\vec{t}_2$ . Figure 5 compares two meshes of the rectangle with a uniform size field  $h = 0.1$ . The first mesh makes use of the cost function (4). The elements of this mesh are mainly oriented with the  $x$  and  $y$  axes. Yet, some patterns of elements are oriented differently. The use of the alternative cost function (7) with a cross field that is aligned with the coordinate axes allows having a quasi-perfect orientation of the mesh. A mix of both cost functions could also be an alternative.

One can use curvature for aligning the quadrilaterals: it is well known that quadrilateral meshes can be aligned in an optimal way for approximating surfaces, depending on the sign of the two eigenvalues of the local curvature tensor. Again, it is possible to find an objective function that takes that into account.

The issue of choosing both an optimal cost function and a suitable cross field is currently under investigation, and we see in these choices many perspectives for further improvements of the Blossom-Quad algorithm.

### 5. EXISTENCE OF PERFECT MATCHINGS

If for some graphs it is possible to find different perfect matchings, there is in general no guarantee that even one single perfect matching exists in a given graph. Consider the meshes of Figure 6. It is obvious that no perfect matching exists for the coarsest one. The following result, known as Tutte's theorem, proves that none of the two meshes of Figure 6 contains a perfect matching.

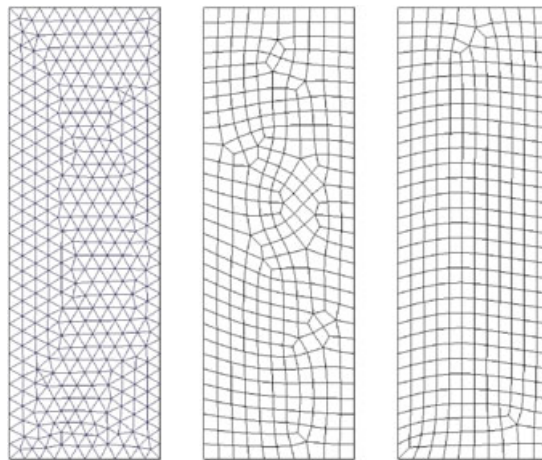


Figure 5. Comparison of meshes generated by Blossom-Quad using cost functions (4) and (7). The initial triangular mesh is on the left. The middle mesh has been generated using (4), and the right mesh uses (7).



Figure 6. Triangulations that have no perfect matching.

*Tutte's theorem:* A graph  $G = (V, E)$  has no perfect matching if and only if there is a set  $S \subseteq V$  whose removal results in more odd-sized components than the cardinality  $n_S$  of  $S$ , that is, the number of elements in  $S$  [16, 17].

In other words, Tutte's theorem says that there is no perfect matching in a triangulation if and only if it is possible to remove  $n_S$  triangles from the mesh and create more  $n_S$  non-connected regions that have an odd-sized number of triangles. Let us use Tutte's theorem to prove that none of the two meshes of Figure 6 has a perfect matching. Let us consider the set  $S$  of triangles that have their tip pointed downwards. In the coarsest mesh,  $n_S = 1$  and 3 odd-sized components are created, which proves that no perfect matching exists. In the second one, six triangles are removed, and 10 odd-sized components are created. This simple pattern can be repeated to produce meshes of arbitrary sizes that have no perfect matchings.

The general problem of counting the number of perfect matchings in a general graph is #P complete.<sup>§</sup> In other words, there is no hope of finding the number of perfect matchings in a general graph. (There is a way to find out, in polynomial time, whether a perfect matching exists by detecting a breakdown in the Blossom algorithm.)

There are however some interesting special cases.

### 5.1. Planar graphs

A graph is said to be planar if it can be drawn in the 2D plane in such a way that its edges intersect only at its vertices. There exists an efficient algorithm (i.e., in polynomial time) that counts perfect matchings in a planar graph. In planar graphs, graph edges form closed non-overlapping loops that form the graph faces. Let  $G$  be a planar graph. Then  $G$  can be oriented efficiently so that each face has an odd number of lines oriented clockwise (this orientation is called a Pfaffian orientation of  $G$ ) [18]. It can be proven that counting the number of perfect matchings can be performed by computing the determinant of the Kasteleyn matrix  $K$ :

$$(\# \text{ of perfect matchings of } G)^2 = \det(K), \quad (8)$$

where the Kasteleyn matrix  $K(G)$  is an adjacency matrix defined as follows. Consider an edge  $e_{ij}$ . If  $e_{ij}$  is oriented positively, then  $K_{ij} = 1$  and  $K_{ji} = -1$ . If  $e_{ij}$  is oriented negatively, then  $K_{ij} = -1$  and  $K_{ji} = 1$ . If no edge exists between  $i$  and  $j$ , then  $K_{ij} = K_{ji} = 0$ .

Here, the computation of the determinant can be carried out in polynomial time so that it is quite easy to count matchings in a triangular mesh. It is therefore possible to compute whether a perfect matching exists in any planar graph. Yet, finding out that no perfect matching exists does not help us a lot at this point. Moreover, the mesh of a whole torus does not lead to a planar graph.<sup>¶</sup>

### 5.2. Cubic graphs

Cubic graphs, also called trivalent graphs, are graphs for which every node has exactly three adjacent nodes. Every cubic graph has at least one perfect matching [19]. It can be proven that the number of perfect matchings in a cubic graph grows exponentially with  $n_V$ .

In a finite element triangulation, most of the triangles of the mesh have three neighbors. Only the triangles that are on the boundary of the domain have less than three neighbors. Thus, in general, a finite element mesh is close to trivalent. We then expect intuitively that perfect matchings will exist in most finite element triangulations. Even though most of the triangulations that we have tried have a perfect matching, the Blossom algorithm has encountered a breakdown for some of the meshes we have tried.

Because cubic graphs always have many perfect matchings, we propose in the Blossom-Quad algorithm to add some extra edges to the graph with the aim of creating a graph topology that is close to trivalent and thus increase the chance of finding perfect matchings.

<sup>§</sup>Sharp P complete, that is, much harder than NP complete.

<sup>¶</sup>The mesh of a complete sphere is planar, even though it is not intuitive.



5.3. Extra edges

In our approach, we propose to add edges (that we call ‘extra edges’ in what follows) in the graph of the triangulation in a way that maximizes the chance of existence of a perfect matching. Consider two successive mesh edges on the boundary of the domain. If those edges are like  $e_1$  and  $e_2$  (Figure 7), their neighboring triangles are not connected in the graph. At this point, we add an extra edge for every pair of triangles that are adjacent to those edges that are successive in the boundary of the domain. Those new connections are represented as dotted cyan lines in Figure 7.

Some of those successive mesh edges correspond to triangles that are already connected, like  $e_3$  and  $e_4$  or  $e_2$  and  $e_5$  in Figure 7. This implies that some of the graph nodes have two neighbors; those ones are rounded in red in Figure 7. Some others have four neighbors: those are rounded in green in Figure 7. Yet, there is no node of the graph that has only one adjacent node.

We have not been able to prove that the graph with those extra edges has always a perfect matching. Yet, in our experience, the addition of those extra edges allows us to find a perfect matching in every example that we have tried.

Technically, we assign a high value (typically 1000) to the ‘badness’ of those extra edges so that an extra edge belongs to the optimal perfect matching only if there exists no perfect matching in the original graph. We propose two manners of post-processing those extra edges when they belong to the perfect matching.

5.3.1. *Edge swap.* The first algorithm is applied essentially when the extra edge connects two triangles  $t_1$  and  $t_3$  that surround one single triangle  $t_3$  (Figure 8). Edges that are in red in the graph belong to the perfect matching. If  $e_{13}$  belongs to the matching, then edge  $e_{24}$  belongs to the matching as well. It is therefore possible to swap the mesh edge that connects  $t_2$  and  $t_4$  and build an all-quad configuration. Note that the concave quadrangle that has been created will be removed through topological optimization (Section 6).

5.3.2. *Vertex duplication.* The second algorithm consists in duplicating the boundary vertex (Figure 9). This algorithm is applied when the two triangles that are connected by the extra edge are surrounded by more than one quad.

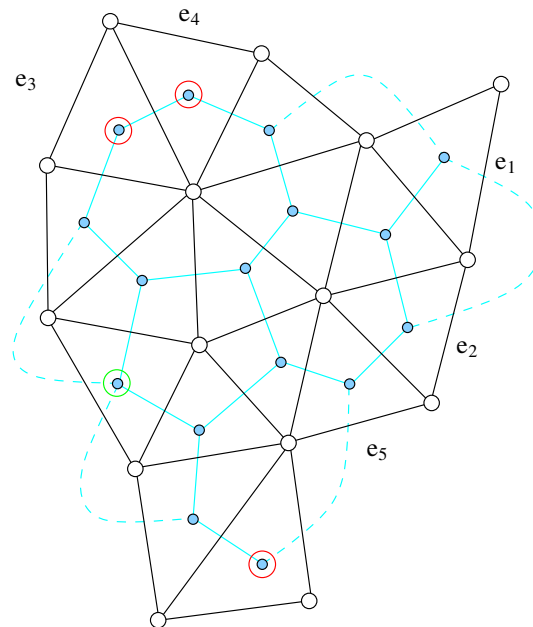


Figure 7. A mesh (in black) and its graph (in cyan). Dotted cyan lines are the extra graph edges that have been added in order to ensure that the graph has perfect matchings.

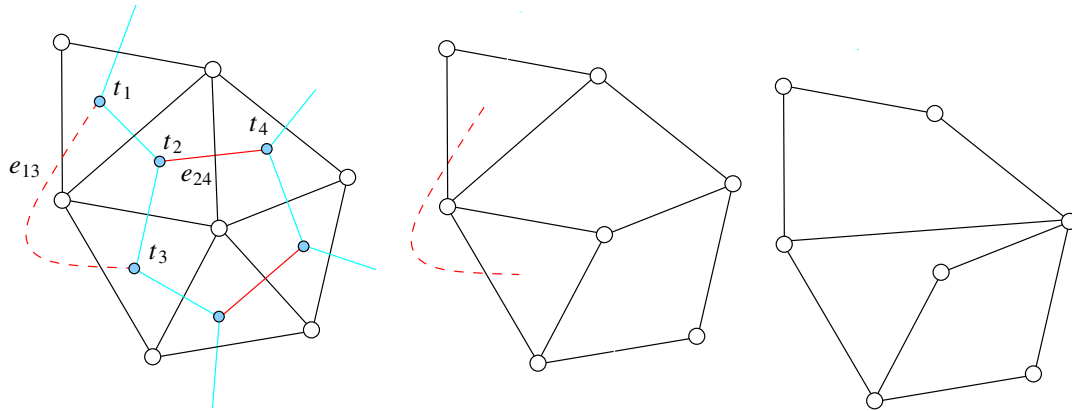


Figure 8. Edge swap algorithm for building an all-quad mesh when an extra edge such as  $e_{13}$  belongs to the matching. Example for a configuration with two triangles and two quads.

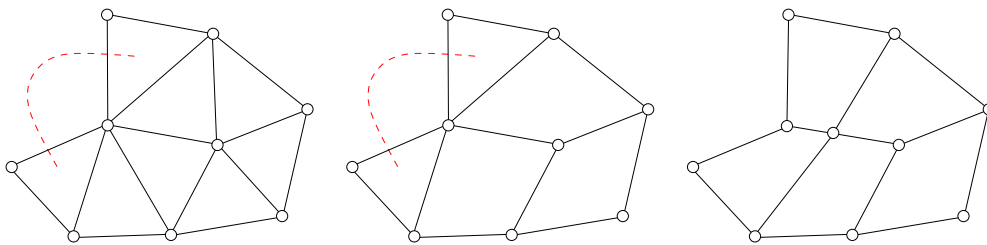


Figure 9. Vertex duplication algorithm for building an all-quad mesh when an extra edge is in the matching.

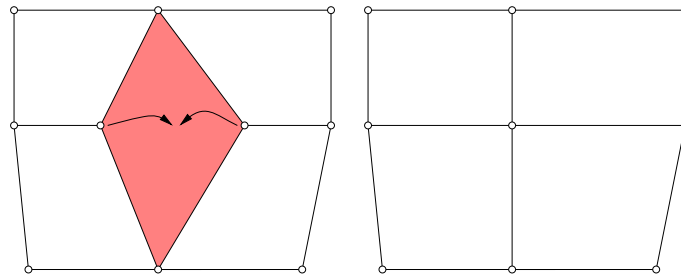


Figure 10. Illustration of quad-vertex-merge optimization operation.

In the next paragraph, we will show how to optimize the quality of the quadrangles of those all-quad meshes using local mesh modifications.

## 6. OPTIMIZATION

In order to enhance the quality of the final mesh, we first apply a standard vertex smoothing procedure [20] to the nodes of the mesh, taking into account the gradation of the size field. Next, we apply two topological optimization operators specifically tailored for quadrilateral meshes.

The topological optimization operators are local deletion operators: a quad-vertex-merge (Figure 10) and the doublet collapse (Figure 11) operation [21]. These operators allow us to remove local mesh structures that have a bad topology. More precisely, the quad-vertex-merge operator replaces two mesh nodes that have three quadrilateral neighbors by one mesh node with four neighbors, and the doublet collapse removes a vertex that has two neighbors.

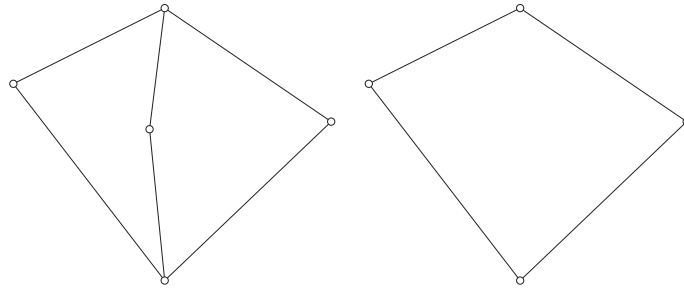


Figure 11. Illustration of a doublet collapse optimization operation.

## 7. THE BLOSSOM-QUAD ALGORITHM

In this section, we summarize the different steps of the new Blossom-Quad algorithm. This algorithm has been implemented in the open-source mesh generator Gmsh<sup>†</sup> [14], and examples of how to use it can be found on the Gmsh wiki.<sup>\*\*</sup>

1. Starting from a solid model with a boundary representation, mesh every model edge  $b_i$  with an odd number of mesh vertices  $n_v^{b_i}$  (6). Then, mesh the model faces with any 2D triangulation algorithm. According to the Euler Equation (5), there will then be an even number of mesh triangles.
2. From the produced mesh, build a weighted graph  $G(V, E, c)$  (Figure 3) where the cost function associated to each graph edge  $c(e_{ij})$  is given either by (4) or by (7). This weighted graph has then an even number of graph nodes, which is a necessary condition for a perfect matching to exist.
3. Enrich the graph with extra edges such as explained in Section 5.3. Those extra edges are given a very high value of cost function:  $c(e_{ij}) = 1000$ .
4. Run the Blossom algorithm to find the perfect matching for the given graph.
5. If the perfect matching contains no extra edges, go to step 6. If it contains some, apply the edge swap algorithm and the vertex duplication algorithm (Section 5.3).
6. Optimize the resulting all-quad mesh as explained in Section 6.

Figure 12 shows the global Blossom-Quad procedure applied to an initial triangular mesh.

## 8. EXAMPLES

In this section, we present the results obtained by applying the new Blossom-Quad algorithm in different contexts. First, we present meshes of simple planar geometries using analytical mesh size fields. Then, we present quadrilateral meshes generated over complex solid models, defined either by a Computer Aided Design (CAD) or StereoLithoGraphy (STL) triangulation. Finally, we show a complex quadrilateral mesh used for multiscale ocean modeling.

### 8.1. Planar quadrilateral meshes with analytical isotropic size fields

This test case has been proposed by Borouchaki and Frey [3]. The domain is a unit square with a circular hole of radius 0.15 centered at (0.75, 0.75). The mesh size field is taken to have a value of  $h(x, y) = 0.003$  along the medial axis of the domain and to have a linear growth from the medial axis to the interior of the domain.

The uniform refinement step that is applied in the recombination algorithm of Borouchaki and Frey [3] has two consequences. On the one hand, it naturally creates a mesh that has a better connectivity. On the other hand, it reduces the efficiency index  $\tau$  of the mesh. As the recombination

<sup>†</sup><http://geuz.org/gmsh/>

<sup>\*\*</sup><https://geuz.org/trac/gmsh/wiki> (username: gmsh and password: gmsh)

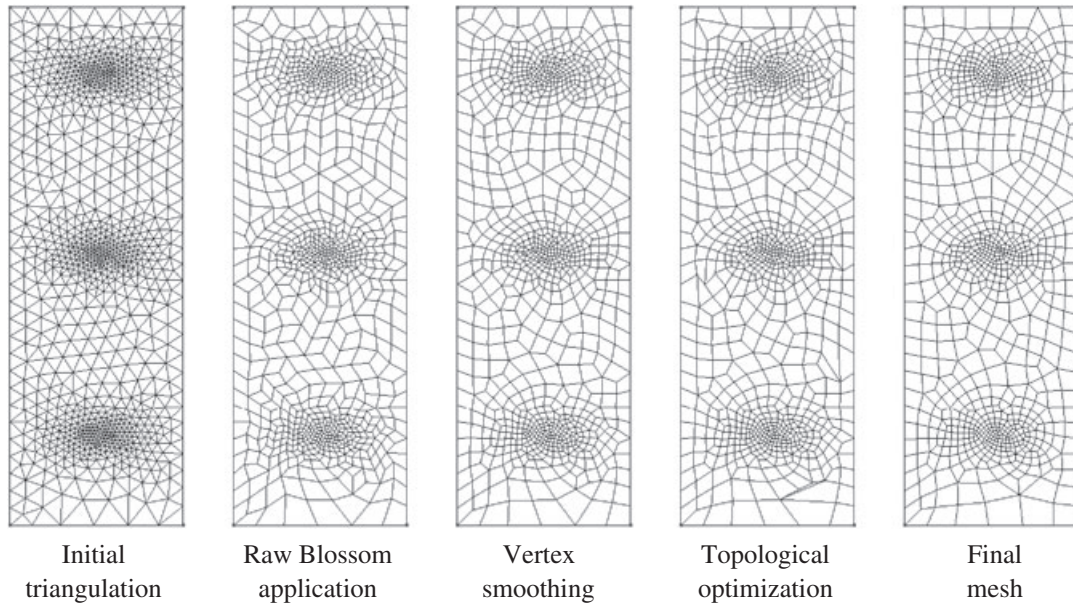


Figure 12. Illustration of the whole Blossom-Quad algorithm.

algorithm of Borouchaki and Frey [3] includes a mesh refinement step, we propose here to generate three meshes with the Blossom-Quad algorithm. The first one applies Blossom to a triangular mesh of size  $h$ . The second one applies the Blossom algorithm to a mesh of size  $2h$ , with one subsequent uniform refinement. The last one applies the Blossom algorithm to a mesh of size  $4h$ , with two uniform refinements.

Figure 13 compares the quad mesh obtained by Borouchaki and Frey [3] and the three meshes obtained with the presented Blossom-Quad algorithm. For the problem with size  $h$ , the Blossom-Quad algorithm takes 1.38 s on a MacBook Pro (Apple Inc., Cupertino, CA, USA) clocked at 2.66 GHz. The new algorithm provides meshes that are of better quality close to the boundaries of the domain. The new algorithm also provides a smoother gradation of the mesh. This is due to the fact that no mesh refinement phase is required in the new algorithm.

Table I compares the quality of the four meshes. We also present some statistics about the degree of the vertices  $d_i$ , that is, the number of quads surrounded by a vertex. The new Blossom-Quad algorithm always produces meshes with better element qualities. The application of the Blossom-Quad procedure to the mesh of size  $h$  is the best in terms of efficiency  $\tau$ . Yet, it has less nodes with the optimal topology, that is, with four neighbors. Applying the Blossom-Quad to a mesh of size  $2h$  seems to be, at least for this test case, a good compromise. The efficiency is still acceptable, and the quality of the mesh is optimum, both in terms of topology and element quality. Using an initial triangular mesh of size  $4h$  does not seem to be a good option, especially in terms of the efficiency  $\tau$ .

## 8.2. Quadrilateral mesh generation applied to STL models

In this section, we present quad meshes for complex solid models represented only by a triangulation in STL format.

The first triangulation represents the solid model of the Stanford bunny model, and the second represents a cerebral aneurysm.<sup>††</sup> The latter triangulation is the output of an image segmentation procedure performed from medical data (computed tomography scan). For the quad-remeshing procedure, we first compute an automatic triangular remeshing procedure based on a conformal parametrization as described in [22] and then run the presented Blossom-Quad algorithm.

<sup>††</sup>The STL triangulation of the aneurysm can be found on the National Institute for Research in Computer Science and Control website, <http://www-roc.inria.fr/gamma/gamma/download>.

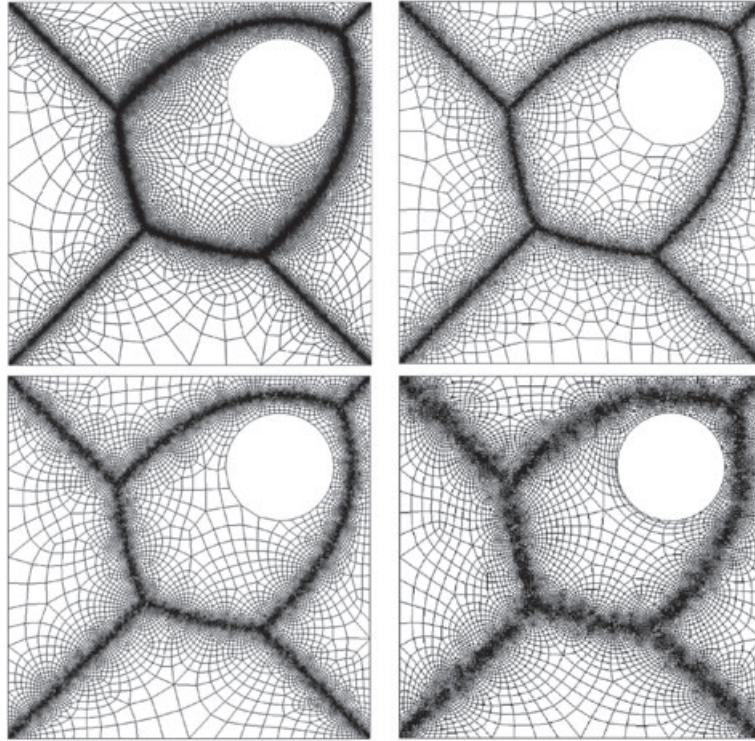


Figure 13. Comparison of both the quadrilateral mesh generation algorithm of Borouchaki and Frey [3] (top, left) and the Blossom-Quad algorithm (other meshes). The top-right mesh has been performed using Blossom on a triangular mesh of size  $h$ . The bottom-left mesh has been generated with the Blossom algorithm applied to a triangular mesh of size  $2h$ , with one subsequent uniform refinement. The last mesh uses the Blossom algorithm on a mesh of size  $4h$ , with two uniform refinements.

Table I. Quality of the quad meshes for the test case with the medial-axis-based mesh size field  $h$ .

Algorithm	Mesh size $h^{\text{algo}}$	Quad quality		Degree vertices			Efficiency $\tau$
		$\eta_w$	$\bar{\eta}$	$d_4(\%)$	$d_{\min}$	$d_{\max}$	
Borouchaki and Frey [3]	$2h$	0.30	0.73	91	3	6	–
Blossom-Quad	$h$	0.32	0.77	72	3	6	85.6%
Blossom-Quad	$2h$	0.39	0.85	94	3	6	82.3%
Blossom-Quad	$4h$	0.31	0.87	98	3	7	75.4%

The algorithms are run with an initial mesh size  $h^{\text{algo}}$  that is subsequently refined to reach the prescribed mesh size field  $h$ . We present values for the minimum quality  $\eta_w$ , mean quality  $\bar{\eta}$ , percentage of vertices of degree 4, minimal and maximal values for the degree of vertices, and efficiency index  $\tau$ .

Figure 14 shows two curvature-adapted remeshed STL surfaces, and Figure 15 presents the quality histograms of those meshes. The curvature-adapted meshes are computed by defining the mesh size  $h(\vec{x})$  as follows:

$$h(\vec{x}) = \frac{2\pi R(\vec{x})}{N_p}, \quad \text{with } R(\vec{x}) = \frac{1}{\bar{\kappa}(\vec{x})}, \quad (9)$$

where  $\bar{\kappa}(\vec{x})$  is the mean curvature that is computed from the initial nodes of the STL triangulation with the algebraic point set surface method (based on the local fitting of algebraic spheres [23]) and  $N_p$  is the number of points chosen for the radius of curvature ( $N_p = 50$ ).

The overall remeshing procedure for both STL examples takes 21 s: 16 s for the automatic triangular remeshing procedure and only 5 s for the blossom-Quad algorithm. The remeshing was

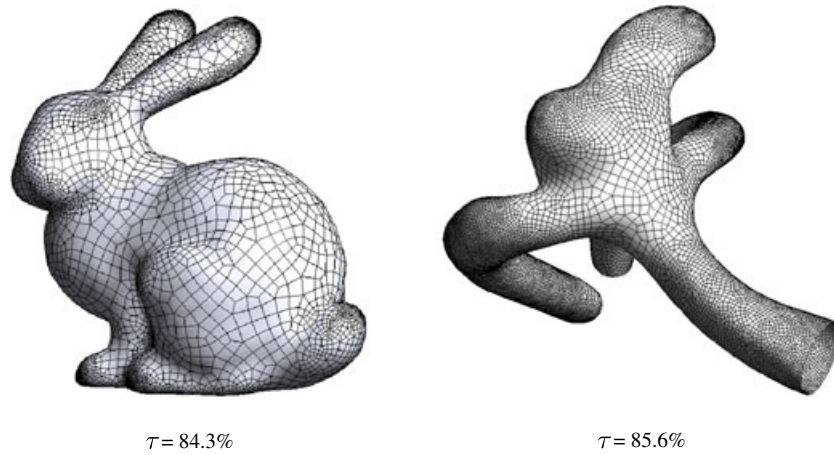


Figure 14. Isotropic mean-curvature-based quad meshes of 13,000 elements for the STL bunny model (left) and 17,000 elements for the aneurysm model (right).

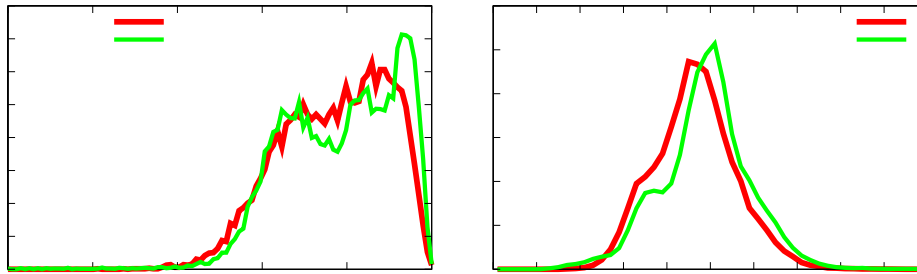


Figure 15. Quality histograms (element quality  $\eta$  (3) and normalized edge length  $l^h$  (1)) for the Blossom-Quad remeshing of the bunny and aneurysm models.

performed on a MacBook Pro clocked at 2.66 GHz. This quad-remeshing procedure is extremely fast compared with the quad-dominant meshes of Lévy and Liu [24] for which the remeshing of the Stanford bunny takes 271 s.

### 8.3. Quadrilateral mesh generation applied to parametric CAD models

We consider the solid model of a human bone. The model contains 11 non-uniform rational B-spline surfaces. The mesh size field is based on surface curvature: we use formula (9) with  $N_p = 35$ . The total time for surface meshing was 26 s for generating 10,134 quadrilateral elements. The time for computing all 11 perfect matchings and doing optimization does not exceed 10 s. The average quality of the finite element mesh is  $\bar{\eta} = 73.3\%$ , and the efficiency is  $\tau = 80.2\%$ . The model still contains one element of bad quality, with  $\eta = 0.01$  (Figure 16).

Gmsh allows direct access to the CAD model, allowing us to compute exactly the principal directions of curvature together with minimal and maximal curvature. This allows us to define an anisotropic mesh size field [7]. Here, we use formula (9) with  $N_p = 35$  in each of the directions of principal curvature. An anisotropic triangular mesh is initially built using the anisotropic metric provided by the curvature. Then, the Blossom-Quad algorithm is applied to it. The resulting mesh is presented in Figure 17. The quadrilateral mesh naturally aligns itself to the principal directions of curvature, allowing us to build an anisotropic quadrilateral mesh without effort.

### 8.4. Quadrilateral meshes for ocean modeling

Our research team has developed the first multiscale hydrodynamic model of the whole Great Barrier Reef. The Great Barrier Reef is on the continental shelf of the Australian northeastern coastline and

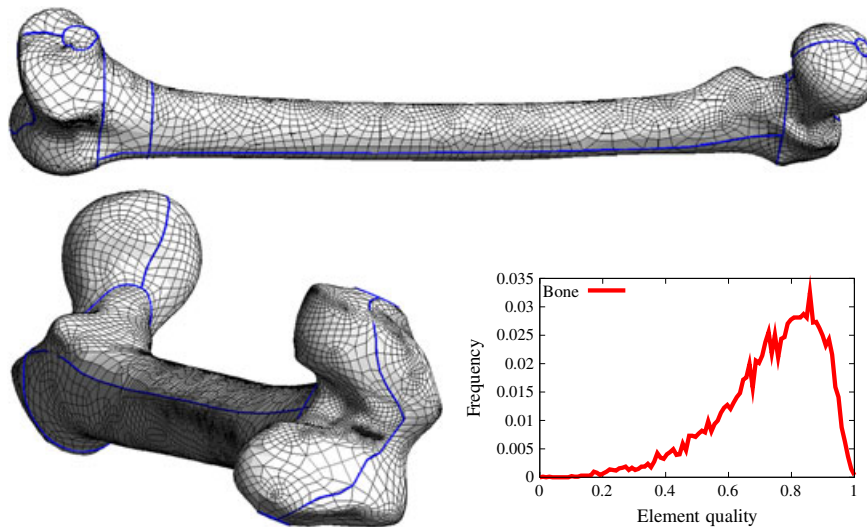


Figure 16. Isotropic quad mesh of the bone geometry with a quality plot.

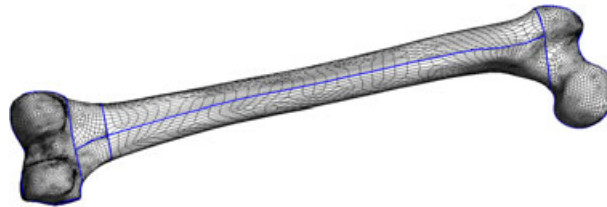


Figure 17. Anisotropic mesh of the bone geometry.

contains over 2500 coral reefs in a strip that is about 2600 km in length and 200 km in width. The mesh size field  $h(\vec{x})$  is defined as a function of the bathymetry and the distance to the shore.

We have performed 24 h of simulations of the water circulation on the Great Barrier Reef shelf. The physical model is described in [25]. The equations are discretized with  $P_1^{DG}$  discontinuous finite elements combined with a second-order multirate explicit Runge–Kutta temporal integrator as in [26]. A plot of the velocity vectors and the sea surface elevation is presented in Figure 18. Tidal jets and eddies due to the interaction of the flow with the topography near the open-sea boundary are clearly visible. The same simulation on a triangular mesh ( $\simeq 250,000$  triangles) was 2.7 times slower than on the corresponding Blossom-Quad quadrilateral mesh ( $\simeq 119,000$  quads).

## 9. CONCLUSIONS

The main contribution of this paper is to have introduced a well-known result of graph theory (the Blossom algorithm for minimum-cost perfect matching) in the domain of unstructured quadrilateral mesh generation. We have presented a new algorithm—dubbed Blossom-Quad—that takes advantage of this result to produce high-quality quadrilateral meshes in a robust and efficient manner, and we have applied it in different contexts: from planar geometries to parametric CAD models to STL remeshing to multiscale ocean models.

Possible further improvements to the proposed algorithm are numerous. For example, the cost function that has been used could be modified in order to align the quadrilateral mesh with some preferred directions. (We have already presented a simple example to that effect in Section 4.3.) Also, in this paper, we have used triangulations with vertices distributed fairly uniformly as input to the recombination procedure. Recent developments [24, 27] allow us to align those vertices with some prescribed directions. Coupling both approaches could potentially lead to even higher-quality

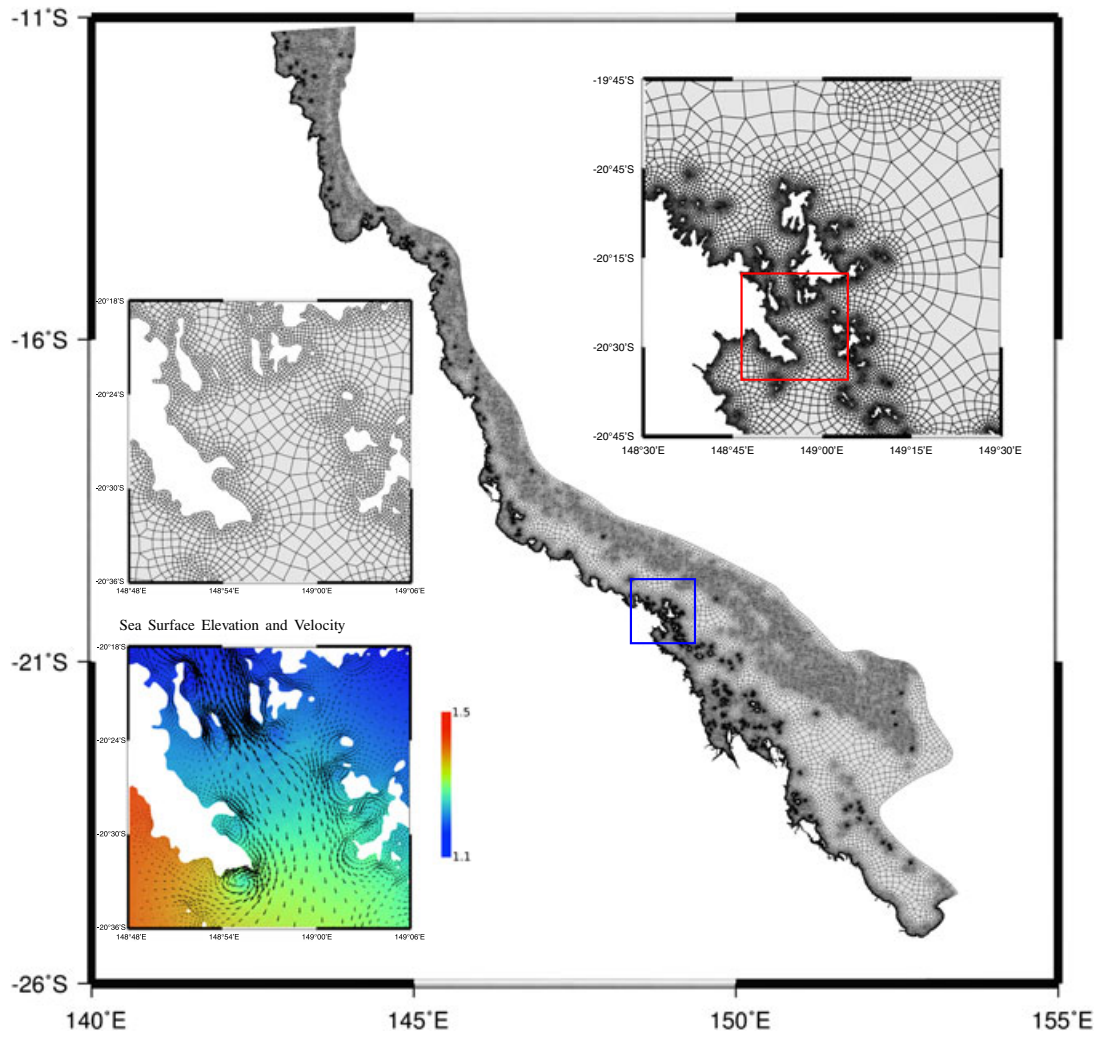


Figure 18. Quadrilateral mesh of the Great Barrier Reef. Two successive zooms of the Whitsunday Islands Archipelago. Sea surface elevation (*color levels*) and bidimensional velocity field (*arrows*).

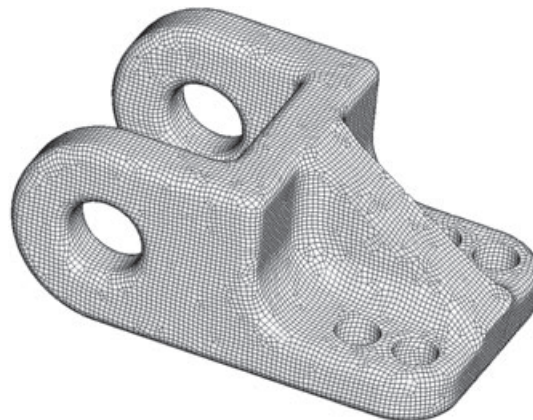


Figure 19. Blossom-Quad algorithm applied to a triangular surface mesh that has been smoothed using the  $L_p$  centroidal Voronoï tessellation technique of Lévy and Liu [24].



quadrilateral meshes. As an example, we have applied the Blossom-Quad algorithm (without optimization) to a triangular surface mesh that has been smoothed using the Lp centroidal Voronoi tessellation technique of Lévy and Liu [24]. The resulting mesh is presented in Figure 19. There, quadrangles do not form patches that are randomly aligned but follow a regular pattern.

In the longer run, the very challenging problem of automatic generation of hexahedral-dominant meshes could be approached using an indirect technique of this kind. In this case, more than two tetrahedra have to be merged in order to form one hexahedron. Here again, we think that graph theory could maybe help us in finding some kind of optimal matching.

#### ACKNOWLEDGEMENTS

This work has been partially supported by the Belgian Walloon Region under WIST grants ONELAB 1017086 and DOMHEX 1017074.

Authors gratefully thank F. Glineur and J. Hendricks from the Applied Mathematics Department of the Université Catholique de Louvain for the discussions and hints about graph theory.

Authors also acknowledge P. Frey of Université Paris VI for authorizing us to include one of the illustrations of his paper [3] (Figure 13, top left).

Authors finally deeply acknowledge B. Levy of LORIA Nancy for providing us with the triangulation and the smoothing algorithm used to produce Figure 19.

#### REFERENCES

1. Blacker TD, Stephenson MB. Paving: a new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 1991; **32**:811–847.
2. Lee CK, Lo SH. A new scheme for the generation of a graded quadrilateral mesh. *Computers and Structures* 1994; **52**:847–857.
3. Borouchaki H, Frey P. Adaptive triangular–quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 1998; **45**(5):915–934.
4. Owen SJ, Staten ML, Canann SA, Saigal S. Q-morph: an indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering* 1999; **9**:1317–1340.
5. Edmonds J. Maximum matching and a polyhedron with 0–1 vertices. *Journal of Research of the National Bureau of Standards* 1965; **69B**:125–130.
6. Edmonds J, Johnson EL, Lockhart SC. Blossom I: a computer code for the matching problem. *Report*, IBM T.J. Watson Research Center, Yorktown Heights, New York, 1969.
7. Frey P, George PL. *Mesh Generation—Application to Finite Elements*. Wiley: Hoboken, 2008.
8. Edmonds J. Paths, trees, and flowers. *Canadian Journal of Mathematics* 1965; **17**:449–467.
9. Lawler EL. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart, and Winston: New York, NY, 1976.
10. Gabow H. Implementation of algorithms for maximum matching on nonbipartite graphs. *PhD Thesis*, Stanford University, 1973.
11. Gabow H, Galil Z, Micali S. An  $O(ev \log v)$  algorithm for finding a maximal weighted matching in general graphs. *SIAM Journal on Computing* 1986; **15**:120–130.
12. Gabow HN. Data structures for weighted matching and nearest common ancestors with linking. *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, 1990; 434–443.
13. Cook W, Rohe A. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing* 1999; **11**(2):138–148.
14. Geuzaine C, Remacle JF. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering* 2009; **79**(11):1309–1331.
15. Bommès D, Zimmer H, Kobbelt L. Mixed-integer quadrangulation. *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*. ACM: New York, NY, USA, 2009; 1–10, DOI: 10.1145/1576246.1531383.
16. Tutte WT. A family of cubical graphs. *Proceedings of the Cambridge Philosophical Society* 1947; **43**:459–474.
17. Pemmaraju S, Skiena S. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica* ®. Cambridge University Press: New York, NY, USA, 2003.
18. Kasteleyn PW. Dimer statistics and phase transitions. *Journal of Mathematical Physics* 1963; **4**:287–293. DOI: 10.1063/1.1703953.
19. Oum S. Perfect matchings in claw-free cubic graphs. *ArXiv e-prints*, Jun 2009.
20. Sarrate J, Huerta A. An improved algorithm to smooth graded quadrilateral meshes preserving the prescribed element size. *Communications in Numerical Methods in Engineering* 2001; **17**(2):89–99.
21. Daniels J, II, Silva CT, Cohen E. Localized quadrilateral coarsening. *SGP '09: Proceedings of the Symposium on Geometry Processing*. Eurographics Association: Aire-la-Ville, Switzerland, 2009; 1437–1444.

22. Marchandise E, de Wiart C, Vos W, Geuzaine C, Remacle J. High-quality surface remeshing using harmonic maps—part III: surfaces with high genus and of large aspect ratio. *International Journal for Numerical Methods in Engineering* 2011; **86**:1303–1321.
23. Guennebaud G, Germann M, Gross M. Dynamic sampling and rendering of algebraic point set surfaces. *Computer Graphics Forum* 2008; **27**:653–662.
24. Lévy B, Liu Y. Lp centroidal Voronoi tessellation and its applications. *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)*, 2010.
25. Lambrechts J, Hanert E, Deleersnijder E, Bernard PE, Legat V, Wolanski JFRE. A high-resolution model of the whole great barrier reef hydrodynamics. *Estuarine, Coastal and Shelf Science* 2008; **79**(1):143–151. DOI: 10.1016/j.ecss.2008.03.016.
26. Constantinescu EM, Sandu A. Multirate timestepping methods for hyperbolic conservation laws. *Journal of Scientific Computing* 2007; **33**:239–278. DOI: 10.1007/s10915-007-9151-y.
27. Hausner A. Simulating decorative mosaics. *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, 2001; 573–580.