3A.1

An Extension of the Reliable Whisker Weaving Algorithm

Franck Ledoux and Jean-Christophe Weill

CEA/DAM Ile de France, DIF/DSSI, BP 12, 91680 Bruyères-Le-Châtel, France
{franck.ledoux@cea.fr|jean-christophe.weill}@cea.fr

Summary. In many FEM applications, hexahedral meshes are necessary to get the best results. However the automatic generation of an hexahedral mesh from an arbitrary 3D geometry still remains a challenge. In this paper, we propose an extension of the reliable Whisker Weaving algorithm [6]. The Whisker Weaving algorithm starts from a pre-meshed quadrilateral surface, it generates hexahedral meshes for a large spectrum of 3D geometries successfully, but it often creates poor-shaped hex elements. We show that considering geometric information in the shrinking loop selection process can provide better results. We introduce criteria to handle non-convex geometries, parallel loops and sheets bounded by several loops.

Key words: hex mesh generation, topology, whisker weaving, parallel loops, geometric properties

1 Introduction

For many years, the finite element method (FEM) has been successfully applied in simulation. Such a method needs a finite element mesh model obtained from a geometric CAD model and suitable for a numerical analysis. It is known that, in many FEM applications, hexahedral meshes give better results than tetrahedral or hex-dominant ones. Last decade showed many attempts to automatically generate hexahedral meshes from arbitrary 3D geometries [14, 2, 18, 16, 6, 5, 10, 15, 13, 12]. However, the automatic generation of hexahedral meshes is more complex than for tetrahedral meshes and still remains a challenge.

For our simulation problems, high mesh quality is required more near the boundary of the solid than deep inside. This property led us to study advancing-front algorithms that we classify in two main categories: algorithms which are essentially based on the geometry and algorithms which focus on the mesh topology. The former have the advantage of capturing the boundary very well but they are sensitive to numerical approximation and they need for

expensive intersection calculations [14]. The latter do not have the drawbacks of the first ones but they often do not provide a satisfying result [6]. Actually, as authors of [6, 9, 13, 12], we think it is essential that any algorithm which attempts to automatically generate hexahedral meshes must take both the model topology and the model geometry into account.

In this paper, we introduce an extension of the reliable Whisker Weaving algorithm [6]. This algorithm has the advantages of being mathematically well-founded and successfully generating hexahedral topology for a wide spectrum of solid geometries. Its principle consists in driving the hexahedral element creation by introducing some complete layers of hexes in one go. The global mesh and model topology are well considered in the reliable Whisker Weaving algorithm but the global geometry may not be. Indeed, geometric information is not considered to select which layer of hexes to create. We suggest to extend the reliable Whisker Weaving by paying attention to the geometric information in the selection process. Moreover, we add new rules in order to manage parallel loops and non-convex regions. Even though we are using geometric information, the algorithm remains mainly topological. Note also that we do not attempt to extend any quadrilateral surface meshes to hexahedral meshes. Following [6, 9, 10], our approach is restricted to quadrilateral surface meshes without any self-intersecting loops in the dual of the surface mesh. Self-intersecting loops seem to be a difficulty to get a robust hexahedral meshing algorithm without degenerated hexahedral elements.

The remainder of this paper is organized as follows. Section 2 reminds of the notion of Spatial Twist Continuum and gives the rough lines of the reliable Whisker Weaving algorithm. Section 3 explains why we need to extend the reliable Whisker Weaving algorithm and thus introduces the concept of *high-level* and *intermediate* rules. We detail these rules respectively in Sects. 4 and 5 while Sect. 6 shows how basic and intermediate rules propagate the geometric information during the overall hex meshing process. A brief summary of this process is given in Sect. 7. Finally, we show some examples in Sect. 8 and we discuss future directions of this work in Sect. 9.

2 Whisker Weaving and Spatial Twist Continuum

The Whisker Weaving algorithm is based on the Spatial Twist Continuum [18], or STC, which structures the dual of a hexahedral mesh into surfaces or *sheets*. This algorithm is an advancing-front algorithm which attempts to address the hex meshing from a purely topological approach. Geometric characteristics are considered secondary to the overall mesh topology. This algorithm first generates the complete dual mesh, from which the primal mesh is obtainable. The spatial location of internal nodes is not computed until the mesh topology is totally defined. The first version of the Whisker Weaving algorithm [18, 16] only considered local element connectivities to create a single hex. This way, the global mesh topology was not taken into account.

The reliable Whisker Weaving version [6] corrects this drawback by introducing a global scheme to create some complete layers of hexes. The principle of the reliable Whisker Weaving algorithm consists in driving the hex creation by selecting a loop of quadrilateral elements on the surface front mesh. A loop of quadrilateral boundary elements corresponds to a layer of internal hexahedral elements. The rules to select the shrinking loop are mainly topological. Unlike the first version of the Whisker Weaving algorithm, the reliable Whisker Weaving algorithm starts from a quadrilateral boundary mesh without self-intersecting loop. This restriction makes it possible to get a robust algorithm that generates less wedges [3].

In the next Section, we begin by reminding the vocabulary relative to the STC before describing the reliable Whisker Weaving algorithm.

2.1 STC: the Fundamental Structure of any Hexahedral Meshes

The STC structures the dual of a hex mesh into *sheets*. Every sheet is dual to a layer of hexes. For example, consider Fig. 1 which shows an hex mesh (on the left) and its corresponding dual mesh (on the right).

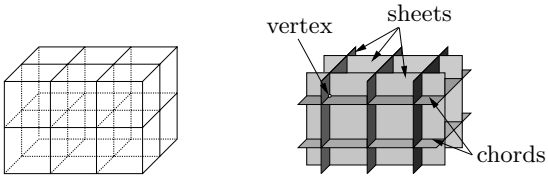

Fig. 1. An hexahedral mesh and the corresponding STC structure

The two green sheets in the STC are dual to the six lower hexes and the six upper hexes in the primal mesh.The intersection of two sheets is a *chord*. A chord is dual to the intersection of two layers of hexes, in other words, a line of hexes. A sheet which intersects oneself is a *self-intersecting sheet*. The intersection of three sheets is a *vertex* which is dual to a hex. In the primal mesh, the intersection of a layer of hexes with the quadrangular boundary mesh is formed by zero, one or more closed lines of quadrilateral elements. Such lines are defined by recursively passing from an edge of a quad to the edge opposite it. The dual of a boundary quad line is a *loop* which bounds a sheet. A sheet with no loop is a *blind sheet* and it contains at least one *blind chord* which is a chord which never meets the boundary.

2.2 A Brief Overview of the Reliable Whisker Weaving [6]

The reliable Whisker Weaving algorithm is based on a loop contraction algorithm which creates a complete layer of hexes in one go. We introduce this

algorithm in this Section, but before we recall the basic weaving operations which are used to perform it.

Basic Weaving Operations

Three basic operations are useful for the reliable Whisker Weaving algorithm[1]: the *cross*, the *seam* (or *join*) and the *blind chord formation* (see Fig. 2). The cross operation creates an hex from three front faces which pairwise share edges. The seam operation completes a chord, i.e. a layer of hexes. In the primal mesh, it consists in joining two faces, which share exactly two adjacent edges, to make one face. The blind chord formation operation creates an hex from only two meshing front faces which share an edge. It introduces a blind chord which traverses the new hex by the two faces which are not opposite to an initial meshing front face in the hex. Note that every operation modifies the meshing front, i.e. the topological loop arrangement. This property is important and will be used in the loop contraction process.

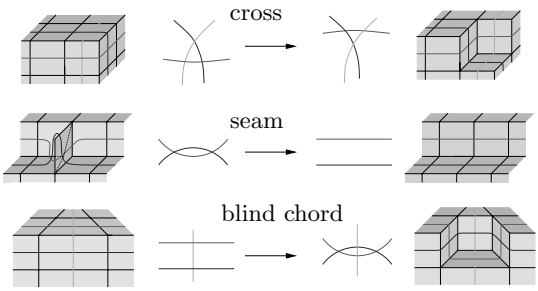

Fig. 2. The three basic rules used in reliable Whisker Weaving

Even though the Whisker Weaving algorithm is a topological advancing-front algorithm, some geometric pre-conditions are defined for every rule. For instance, the creation of an hex from three boundary faces is possible only if the angles between the faces are not too flat.

Loop Shrinking

The reliable Whisker Weaving algorithm is based on the observation that contracting a curve to a point on a sphere sweeps out a surface behind it inside the ball [7]. For this algorithm, the curves are the loops on the dual surface and the swept out surfaces are the sheets. In the primal mesh, shrinking a loop creates a complete layer of hexes in an advancing-front fashion. The

[1] Note that other rules are defined in the initial Whisker Weaving algorithm [16].

algorithm chooses which loop to shrink and decides to shrink it left or right. This operation is repeated until all the loops are shrunk.

Once a direction is selected, shrinking a loop consists in crossing it with all the other loops it came across from until it no longer intersects other loops. At this time the loop shrinks to a point and disappears. This process can also be interpreted as collapsing the faces which are adjacent to the loop in the shrinking direction. Collapsing a face consists in performing some basic weaving operations to reduce its number of vertices until it totally vanishes. For instance, consider the 5-sided face on Fig. 3. The red curve is the curve to shrink. The first operation to apply is a blind chord formation. It reduces a k-sided face to a $k-1$-sided face. In the example, the 5-sided face becomes 4-sided. A second blind chord formation transforms the gray face in a 3-sided face. Applying a cross operation completes the cell collapsing.

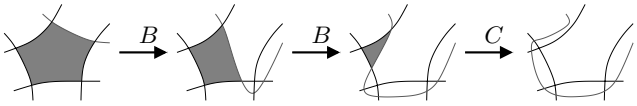

Fig. 3. Collapsing a cell

Now consider the overall shrinking process with Fig. 4 where we want to shrink the black loop inward. For every face adjacent to the shrinking loop, a figure indicates its number of sides. The idea is to begin by traversing once about the loop to find a face with the fewest number of sides and collapse it. Then we look again for the face with the fewest number of sides and we collapse it. This process is repeated until the loop shrinks to a point.

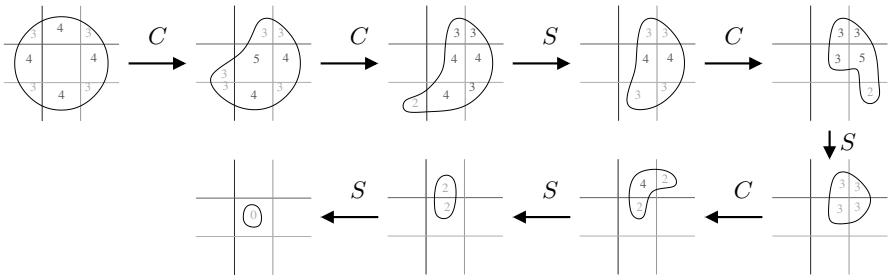

Fig. 4. An example of shrinking process

The authors of [6] claim that the best order to select the shrinking loop is by increasing *weight*. Each loop has two weights, corresponding to shrinking it left or right. Let n_{rp} be the number of points on the right of the curve, n_{lp} the number of points on the left of the curve and n_{pc} the number of points on

the curve. The left weight is defined as $w_l = n_{lp} + n_{pc}$ and the right weight as $w_r = n_{lp} + n_{pc}$. The best choice is to select the loop with the smallest weight in a direction (left or right). This pure topological approach is balanced by introducing some geometric constraints. Actually, local rules associated with basic weaving operations are blended into the decision structure of the loop shrinking process. These rules are either geometric, which only apply to hexes near the model boundary, or topological, which apply universally. It seems that local rules are not used to select the shrinking loop but just to help drive the loop shrinking process. Note that, according to the authors of [6], it is usually necessary to turn the geometric rules off to complete the weave, but rarely necessary to turn the topological rules off.

3 Our Motivations to Extend the Reliable Whisker Weaving

The Whisker Weaving algorithm has the advantage of being mathematically well-founded and of successfully generating hexahedral topology in many cases. Still, generated hexes are often poorly shaped or even inverted. Two reasons could explain this drawback. First, the result of the Whisker Weaving algorithm strongly depends on the initial boundary mesh, this drawback is common to all meshing algorithms which start from a fixed boundary mesh. Second, the choice of the layer of hexes to introduce is only driven by topological information while geometric information may be useful. For instance, consider Fig. 5 which shows five topologically equivalent surface meshes. All these meshes have 14 faces, 28 edges and 16 vertices.

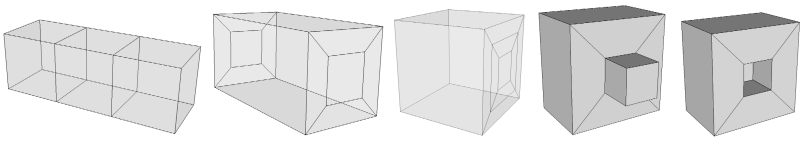

Fig. 5. Five different solids bounded by the same topological quad mesh

An obvious solution to mesh the first boundary surface mesh is to introduce three hexes. However this solution is not suitable for the other cases while they are topologically equivalent. A careful loop selection process has to take the surface geometry into account. Possible results obtained with our approach are shown on Fig. 6. To get such results, we consider geometric properties like convexity and particular topological configurations like parallel loops. These examples are quite typical of our concerns considering that we use this algorithm to create blocks in an hex meshing block-structured approach.

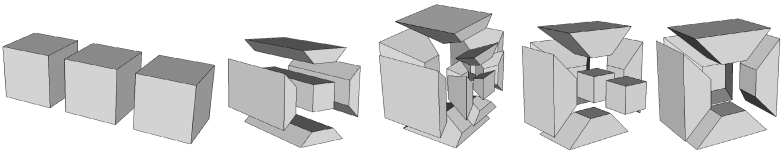

Fig. 6. Hex meshes which correspond to the surface meshes shown on Fig. 5

Our choices have led us to provide an algorithm built on three rule sets. First, we define *high level* rules to select which loop to shrink. These rules pay attention to geometric and topological information. Then, *basic rules* which are equivalent to the basic weaving operations are used to shrink a loop. Finally, we introduce *intermediate level rules* between the high level rules and the basic rules. They locally modify the arrangement of loops by introducing hexes or applying particulars seams. These local changes are driven by global geometric and topological considerations.

4 High-Level Rules to Drive the Shrinking Loop Selection

We are currently using five high-level rules which are useful for selecting the loop to shrink. They depend on some properties relative to each loop. As a loop can be shrunk left or right, each property is actually associated with a loop and a direction. Keep in mind that a difference between our approach and the Whisker Weaving algorithm is that we work on the primal mesh and not on the dual mesh. The convex rules (see below) and the properties necessary to these rules are the same ones as those in [10].

4.1 The Depth Rule

In many numerical applications, high mesh quality is required closer to the boundary than inside the mesh. In order to enforce that important characteristic, we defined the *depth rule*. It requires a value is assigned to each face, this depth value being smaller when the face is closer to the boundary.

A left depth index d_l and a right depth index d_r are associated with each loop. For a direction i, we consider all the faces on the i side of the loop and we define d_i as being equal to the least strictly positive depth index assigned to one of these faces. The depth rule consists in selecting the least depth loops.

4.2 The First Convex Rule

The *first convex rule* consists in selecting the loops that are not lying on a plane surface mesh. This rule attempts to keep the algorithm from building a sheet from one loop when it should be bounded by two loops at least.

In order to know if a loop is lying on a plane surface mesh, geometric information has to be assigned to the boundary. For each edge, we determine the dihedral angle α between the two quadrilateral faces which share it. Thus each edge is classified as being *sharp* if $\alpha \leq 120$, *a little sharp* if $120 < \alpha < 170$ and *flat* if $170 \leq \alpha \leq 180$[2]. Starting from an edge, a single loop l can be built by recursively passing from an edge of a q quad to the edge opposite it in the q quad until the initial edge is reached. For instance, on the left of Fig. 7, starting from the e edge, the red loop is created. The edges of the quads crossed by the loop are separated in three sets: the set se_c^l contains the edges crossed by the loop l, the set se_l^l contains the edges on the left (the green edges on Fig. 7 left) and the set se_r^l contains the edges on the right (the blue edges on Fig. 7 left). In order to know if a loop is lying on a plane surface mesh, we count the number of sharp and a little sharp edges in the set se_c^l. The loop can be shrunk only when this number is positive.

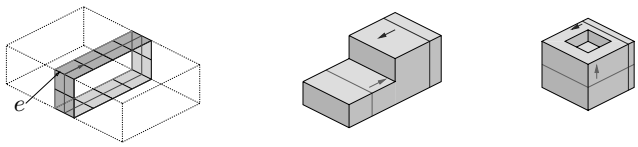

Fig. 7. Left: The three sets of edges associated with a loop. Middle and right: Two non-convex solid geometries and some unacceptable loops shrinkings

4.3 The Second Convex Rule

The *second convex rule* first selects the loops which are along a sharp region of the solid. As [10], we assign a *side elimination weight* to each loop for both directions. For the right side (respectively the left side), this weight is the number of sharp edges in the set se_r^l (respectively se_l^l) divided by the loop length. The second convex rule consists in selecting the highest weight loops.

4.4 The Non-Convex Rule

The goal of the *non-convex* rule is to keep the algorithm from shrinking a loop towards a non-convex part of the solid. For instance, on Fig. 7 (middle and right), red and blue loops can not be shrunk in the indicated directions.

In order to do that, we consider the fact that a non self-intersecting loop separates the edges of the boundary mesh in two sets, se_{gr}^l and se_{gl}^l, which respectively contain all the meshing front edges on the right side and on the left side of the l loop. A loop can be shrunk to direction d only when

[2]Note that the limit values can be modified.

the set sel^l_{gd} contains no non-convex edges. In order to manage non-convex property, we extend the edge classification introduced in Sec. 4.2. Consider that boundary face normals are directed to the outside of the solid, we classify each boundary edge e as a function of the dihedral angle α between the two faces which share it (see Fig. 8): if $\alpha \in]0, 120]$, the edge e is *sharp convex*; if $\alpha \in]120, 170]$, the edge e is *a little sharp convex*; if $\alpha \in]170, 190]$, the edge e is *flat*; if $\alpha \in]190, 240]$, the edge e is *a little sharp non-convex*; if $\alpha \in]240, 360[$, the edge e is *sharp non-convex*. The non-convex rule indicates that it is impossible to shrink a loop right (respectively left) if the number of non-convex edges in se^l_{gr} (respectively se^l_{gl}) is positive.

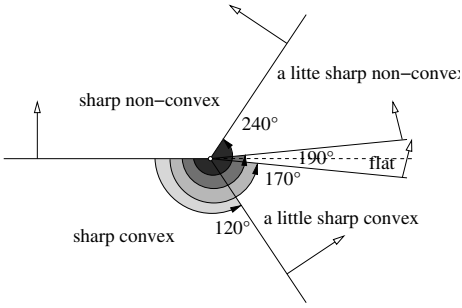

Fig. 8. Edge classification as a function of the dihedral angle

4.5 The Parallel Crossing Rule

Parallel loops have to be considered with attention in an hex meshing algorithm which starts from a fixed quad boundary mesh. This fifth rule has been added to manage the parallel loops separated by a sequence of sharp convex edges. Such configurations appear in some cylindrical mesh parts. For example, consider Fig. 9 which was already introduced on Figs. 5 and 6.

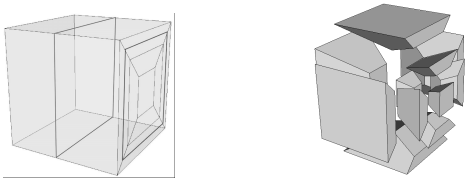

Fig. 9. A configuration where it is necessary to apply the parallel crossing rule

The meshing process must be driven so that it avoids to create a large hex which would completely fill in the geometry. Thus we first consider the blue

and red loops which are topologically parallel and separated by a sequence of sharp convex edges. The idea is to introduce a layer of hexes in such a way that the first created chord is a torus of hexes adjacent to both blue and red loops. This result is obtained by shrinking one of the parallel loops in the direction of the other loop. We currently select the shrinking loop as being the loop which collapses the smallest number of faces when it is shrunk in the direction of the other loop.

5 Intermediate Rules to Locally Modify the Mesh

In this Section, we introduce the intermediate rules which clean up the mesh by applying a local topological modification before selecting a new loop to shrink. They do not focus on a single loop but consider the global topology and geometry. Currently there are only two intermediate rules. But we can legitimately think that this number could grow to improve the algorithm. Both rules are relative to the sheets which are bounded by two loops at least.

5.1 The Parallel Merging Rule

This rule is equivalent to the *double loop elimination* process introduced in [10]. It is efficient for cylindrical-shaped meshes when some sheets are bounded by two loops. More generally, this rule makes it possible to join certain configurations of two sheets into one sheet. Suppose that three loops are parallel and next to each other. Let l be the central loop, l_r be its right parallel loop and l_l be its left parallel loop. If, in addition, all the edges separating l and l_r on the one hand, and l and l_l on the other hand, are classified as sharp edges, the solution consists in eliminating loops l_r and l_l simultaneously. In [10], this elimination is interpreted as removing a chord which forms a torus of hexes, we choose rather to add the *parallel merging rule* shown on Fig. 10 which merges the two parallel loops l_r and l_l in one single loop. Afterward, we shrink this new loop to create a torus of hexes.

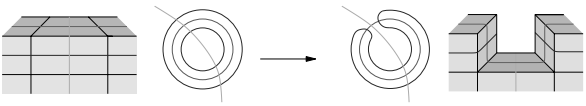

Fig. 10. Merging of two parallel loops

5.2 The Non-Convexity Seam Rule

The basic seam is performed when two faces share exactly two adjacent edges. It joins these faces to make one new face. As a consequence, it finishes creating

a chord of hexes. In this Section, we introduce a new seam rule, called the *non-convexity seam rule*, which joins two faces sharing only one edge when particular geometric conditions are satisfied. This rule completes some chords in non-convex geometries. More precisely, when a chord arrives near a boundary face in a non-convex region, it may be better to directly join the chord and the boundary face. Without this rule, basic weaving rules introduce too many hexes in the final mesh and sometimes fill in non-convex areas.

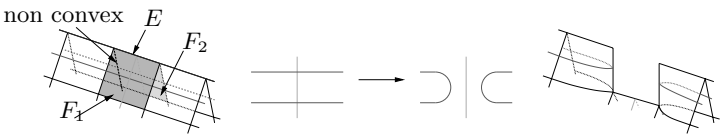

Fig. 11. Example of non-convexity seam

Consider Fig. 11 where F_1 and F_2 are two faces which share[3] the single edge E. There are three possibilities: both faces are boundary faces, both faces are internal free faces or one face is a boundary face and the other one is an internal free face. If F_1 and F_2 are both boundary faces, they can not be sewed considering that their geometric positions are fixed. The case of both internal faces is out of the scope of the non-convexity rule. Indeed the goal of this rule is to complete chords which arrive near a non-convex boundary region. Actually, the non-convexity seam rule is only performed in the third case when a face is on the boundary and the other one is free. Let F_2 be the boundary face and F_1 be the internal free face. Some conditions have to be verified. The edge E has to be a convex[4] edge and an edge of F_2 adjacent to E has to be non-convex. Moreover the two vertices of F_2 which not belong to E must be free internal nodes. Under these conditions, the non-convexity seam rule can be used as illustrated on Fig. 11. Afterward, some faces adjacent to F_1 and F_2 share two edges. Such a degeneracy is removed with the basic seam.

6 Propagation of Geometric Information

The high-level and intermediate rules are based on some geometric properties of the initial surface mesh. These properties could be disposed of when the meshing front is no longer near the boundary, but we prefer to propagate them inside the mesh. This choice allows us to keep internal layer of hexes as parallel to the surface mesh as possible. Consider Fig. 12 where the sharp convex edges and the sharp non-convex edges are respectively colored in red

[3] If F_1 and F_2 share two edges, the basic seam is performed.

[4] If the edge E is convex by propagation, some tests are performed depending on its initial state and the number of hexes sharing it.

and green. We start from the left surface mesh. Creating a first layer of hexes modifies the meshing front. We propagate the geometric properties from the previous meshing front to the new one. In particular some front edges are now defined as being sharp and convex while they were not beforehand. Creating the second layer of hexes shows another characteristic of the propagation process: a convex property and a non-convex property cancel each other out. As a result, when a sharp convex property is propagated onto a sharp non-convex edge, this edge becomes flat.

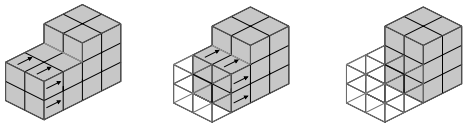

Fig. 12. Propagation of geometric constraints during the meshing process

Since the basic and intermediate rules modify the surface meshing front, they have to propagate the geometric properties. The first property to propagate is the depth. At the beginning, the depth of every boundary face is equal to zero. When a rule is performed, either an hex is introduced or two faces are sewed. In the first case, each new face belongs to the new hex. Let F_1 be such a face and F_2 its opposite face in the hex. If F_2 is a former front face whose depth is equal to i, the depth of F_1 is equal to $i+1$. If F_2 is a new front face too, the depths of F_1 and F_2 are equal to -1. A seam introduces no face, so we do not have to propagate the depth parameter.

Consider Fig. 13 where the red line represents the shrinking loop. The cross rule introduces a hex and then replaces three front faces with three new ones. As explained above, for an initial i-depth face, the corresponding new face depth (opposite in the created hex) is $i+1$. The seam only removes faces. The blind chord formation rule introduces an hex and replaces two faces with four faces. Two new faces correspond to the initial ones. The two other faces are not topologically parallel to a front face. Their depth values are arbitrary fixed to -1. Note that when an initial front face is -1-depth, the parallel face is -1-depth too. The merging rule is similar to the blind chord formation rule and the non-convexity seam rule does not introduce any new faces.

The second property to propagate is the angle and the convex type[5] associated with every edge. As the basic rules are used to shrink a loop, they have to consider the shrinking direction to propagate the angle. Indeed, shrinking a loop creates a layer of hexes and the meshing front is then translated from one side of this layer to the other side. As a result, the basic rules just propagate the angle in a direction as illustrated on Fig. 14. The intermediate rules which are not associated with a shrinking loop, act quite differently. The par-

[5] convex or non-convex.

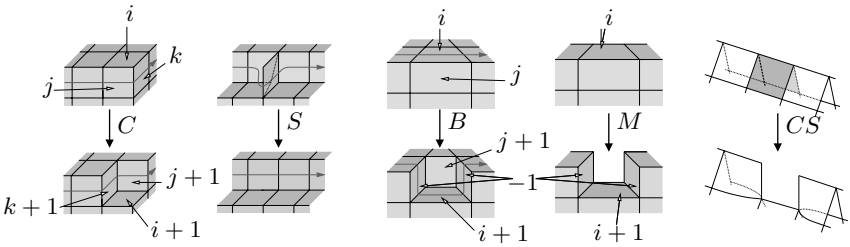

Fig. 13. Rules and depth

allel merging rule propagates the properties associated with the edges which separate the central loop and the parallel loops. The non-convexity seam rule propagates the properties associated with the edge which is shared by the two faces to sew.

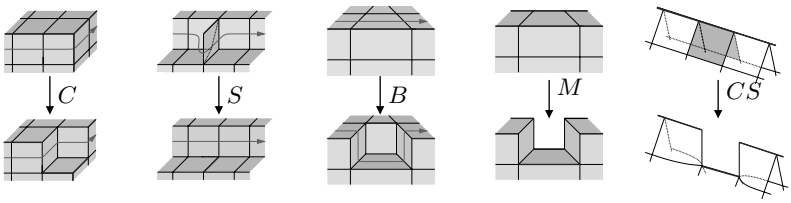

Fig. 14. Rules and convexity

7 A Brief Summary of the Algorithm

The algorithm is described in this Section and illustrated on Fig. 15. It essentially defines in which order the high level rules are applied.

The first step consists in initializing geometric properties. They are computed at the first time and then they are propagated inward without any other geometric computations.

The second step consists in applying the non-convexity rule when it is necessary. This rule is the only one which does not help to choose which loop to shrink. Note that the parallel merging loop rule which is an intermediate rule too implies to shrink the merged loop. This way, it contributes to choose which loop to shrink.

Selecting the shrinking loop needs to compute every loop properties. Actually we compute the properties of every couple (l, d) with l a loop and d a direction. Let S be the set of these couples. For a loop l, S contains the two couples (l, l) and (l, r) which correspond to shrinking l left or right. Then the parallel merging rule and the high-level rules allows us to select which loop

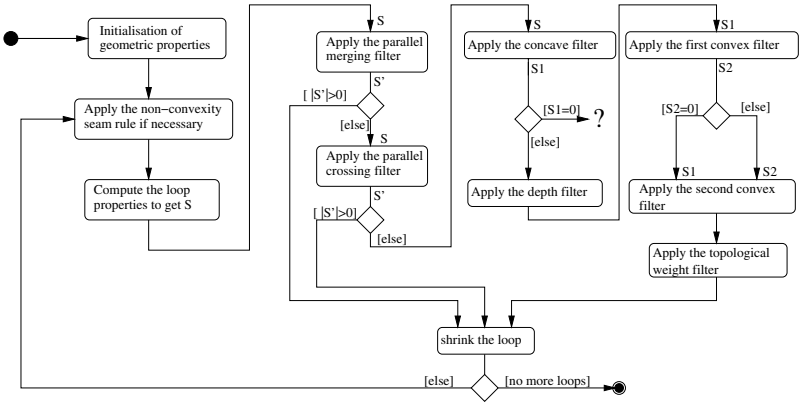

Fig. 15. Operation Sequence to shrink loops

to shrink. We first apply the rules relative to the parallel loops. These rules directly define the loop to shrink and the shrinking process can start. Otherwise, the high-level rules, except the parallel merging rule, are used. Applying such a rule r is equivalent to filter the set S in order to only keep the couples (l, d) which verify the r rule. If S becomes void, we relax some rules.

- The first filter is based on the non-convexity rule. The case when none of the couples (l, d) verifies this rule is currently not supported. We have not experienced yet this very particular case.
- The second filter is based on the depth rule. It keeps the couples (l, d) with the smallest depth. Then, among these couples, we select the couples (l, d) which verify the convex rules.
- The third and fourth filters are based on the first and second rules respectively. The first convex filter may be relaxed.
- The last filter is topological. It keeps the loops which collapse the smallest number of faces.

Note that only two filters can void S out. Once a loop is selected, it is shrunk by performing the same shrinking process as in the Whisker Weaving algorithm. Note that our basic rules have the same kind of pre-conditions as in the reliable Whisker Weaving algorithm. These pre-conditions can be turned off too.

8 Examples

The following examples illustrate the qualities and drawbacks of our approach. The first example on Fig. 16 shows the algorithm ability to work with non-convex geometries and to create sheets bounded by several loops.

The surface mesh of the examples on Figs. 17 and 18 are obtained by applying a paving-like algorithm [4], followed by a sheet removing algorithm [6]

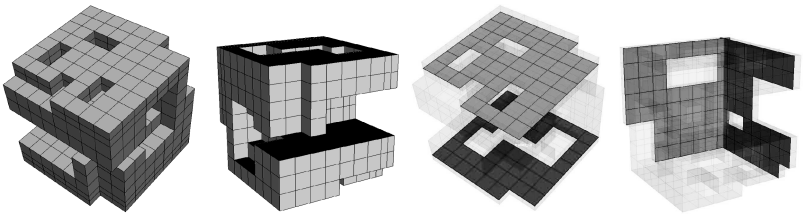

Fig. 16. View of a non convex mesh and several sheets bounded by more than one loop in its dual

coupled with pillowing [8]. This process guarantees to get a surface mesh without any self-intersections. Our algorithm is then used to generate hexahedral meshes. Taking care about geometric information does not prevent our algorithm from generating hexes with negative jacobians at nodes, making them unusable for most FEMs. In fact, far from boundary, topological rules have priority and even if we propagate geometric information inward our algorithm acts like the reliable Whisker Weaving one. So it has the same drawbacks. Note that we have not used any topological smoothing [6] yet to get a "better" mesh. Most of the meshes we have generated have an accumulation point which contains the last introduced layer of hexes by our algorithm. In fact, these meshes have too many sheets. Such a result tends to prove that geometric seams are necessary to reconnect some sheets and so decrease the number of sheets.

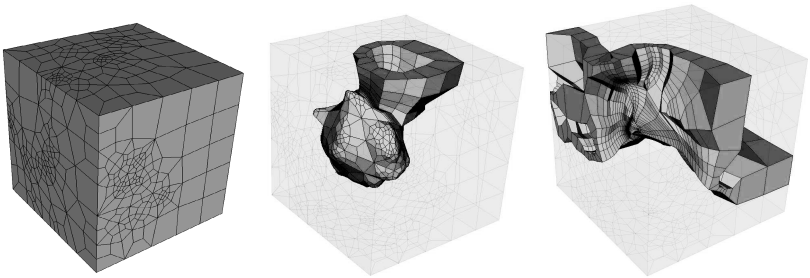

Fig. 17. View of a cube with a complicated surface mesh. It has 14022 elements and 681 with negative jacobians at nodes. The left picture shows the front mesh faces. The middle and right pictures show two layers of hexes. The former surrounds an accumulation point in the mesh while the latter goes through it

[6] In [6], degeneracies like *through-cells*, *non-simplicial meets* and *non-distinct sup-cells* are removed to get a well-defined hex mesh.

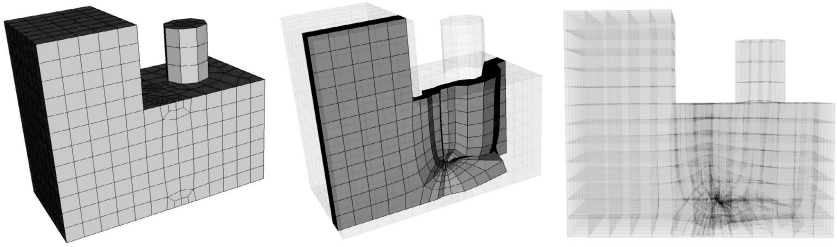

Fig. 18. Another mesh example. It has 2030 elements and 21 with negative jacobians at nodes. The left picture shows the front mesh faces. The middle picture shows a layer of hexes. The right picture shows the internal mesh structure where an accumulation point is visible

9 Future Work

The work presented in this paper is still in progress. Several main directions are identified to complete it. First we have to improve the algorithm robustness by testing it on a wider spectrum of geometries. Especially, we will test it on sparsely meshed boundary surfaces in order to build blocks in a block-structured meshing approach. We will also have to work on the case when the non-convexity filter indicates that none of the loops can be shrunk. A solution should be to perform an incomplete shrinking process which shrinks the loop until it reaches non-convex edges. The non-convexity seam or a new intermediate rule could then clean up the mesh. Such a solution would be useful for managing the parallel loops too. Indeed, in our algorithm, these loops have priority over the other ones and sometimes the selected shrinking loop can be shrunk towards a non-convex area. To get a more general algorithm, we could also attempt to manage self-intersecting loops. That would probably mean to deeply modify our algorithm which intensively uses the fact that a loop cuts the meshing front in two parts (left and right).

As illustrated by the previous Section, both topology and geometry are necessary to get a valid mesh. So the next step of this work will be to adapt our algorithm to get a classical geometric advancing-front algorithm which would be driven by both topological and geometrical rules. In such an algorithm, topological rules must have priority. As our algorithm works on the primal mesh directly, it can be easily extended in such a way.

The efficiency, the robustness and the result of an advancing front method starting from a pre-meshed boundary are strongly connected to the boundary mesh quality. However the surface mesh is generally created without considering that it will be the starting point of a hex mesh generation algorithm. In order to mesh the boundary of a 3D geometric solid, the traditional way consists in meshing its geometric boundary curves then meshing independently its geometric boundary surfaces with a robust quad meshing algorithm like the Paving algorithm [4] or the Q-Morph algorithm [11]. To get an acceptable

boundary mesh for hexahedral meshing, we think that the boundary surfaces have to be meshed together considering the global topology and the geometry of the 3D object. It would be equivalent to create the loops on the boundary geometry in an element by element fashion. The loop could then be shrunk in their creation order.

It is also important to work on the optimization step. If the Whisker Weaving algorithm has the advantage of successfully generating hexahedral topology for a wide spectrum of solid geometries, it has the drawback of often creating nodes which are connected to a great number of hexes. Our algorithm has the same drawback. To improve the quality of hexes, local topological changes have to be performed. Few papers [17, 1] talk about this research field which is yet important. The main difficulty is that local re-meshing in hexahedral mesh can have severe global repercussions.

10 Conclusion

In this paper, we present an extension of the reliable Whisker Weaving algorithm. The new algorithm pays more attention to the geometric information to select the shrinking loop. It also attempts to manage non-convexity areas and parallel loops. The implementation of this algorithm has provided first results but it deserves to be enriched by introducing an incomplete loop shrinking process which seems to be a solution to manage both non-convex areas and parallel loops. Note that the algorithm structure built on different levels and rules with specific filters eases the addition of new rules.

Acknowledgments

The authors wish to thank people who shared their opinions and advices about the preliminaries of this work presented at the IMR'2006 Poster session.

References

1. M. Bern and D. Eppstein. Flipping Cubical Meshes. *ACM Computer Science Archive*, 2002. http://www.arXiv.org/cs.CG/108020.
2. T.D. Blacker and R.J. Meyers. Seams and Wedges in Plastering: A 3D Hexahedral Mesh Generation Algorithm. *Engineering with Computers*, 9:83–93, 1993.
3. T.D. Blacker, S.A. Mitchell, T.J. Tautges, P. Murdock, and S.E. Benzley. Forming and Resolving Wedges in the Spatial Twist Continuum. *Engineering With Computers*, 13:35–47, 1997.
4. T.D. Blacker and M.B. Stephenson. Paving: A New Approach to Automated Quadrilateral Mesh Generation. *International Journal for Numerical Methods in Engineering*, 32:811–847, 1991.

5. N.A. Calvo and S.R. Idelsohn. All-hexahedral element meshing: Generation of the dual mesh by recurrent subdivision. *Computer Methods in Applied Mechanics and Engineering*, pages 371–378, 2000.
6. N.T. Folwell and S.A. Mitchell. Reliable Whisker Weaving via Curve Contraction. *7th International Meshing Roundtable*, pages 365–378, 1998.
7. S.A. Mitchell. A Characterization of the Quadrilateral Meshes of a Surface Which Admit a Compatible Hexahedral Mesh of the Enclosed Volume. In Springer, editor, *13th Annual Symposium on Theoretical Aspects of Computer Science*, pages 465–476, 1996.
8. S.A. Mitchell and T.J. Tautges. Pillowing Doublets: Refining a Mesh to Ensure that Faces Share at Most One Edge. *4th International Meshing Roundtable*, pages 231–240, 1995.
9. M. Muller-Hannemann. Shelling Hexahedral Complexes for Mesh Generation. *Journal of Graph Algorithms and Applications*, 5(5):59–91, 2001.
10. M. Muller-Hannemann. Quadrilateral surface meshes without self-intersecting dual cycles for hexahedral mesh generation. *Computational Geometry*, 22:75–97, 2002.
11. S.J. Owen, M.L. Staten, S.A. Canann, and S. Saigal. Q-Morph: An Indirect Approach to Advancing Front Quad Meshing. *International Journal for Numerical Methods in Engineering*, 44:1317–1340, 1999.
12. M.L. Staten, R.A. Kerr, S.J. Owen, and T.D. Blacker. Unconstrained Paving and Plastering: Progress Update. *14th International Meshing Roundtable*, pages 469–486, 2006.
13. M.L. Staten, S.J. Owen, and T.D. Blacker. Unconstrained Paving and Plastering: A New Idea For All Hexahedral Mesh Generation. *14th International Meshing Roundtable*, pages 399–416, 2005.
14. M.B. Stephenson, S.A. Canann, and T.D. Blacker. Plastering: a new approach to automated 3D hexahedral mesh generation. *Progress Report*, 1992. SAND89-2192.
15. T. Suzuki, S. Takahashi, and J. Shepherd. An Interior Surface Mesh Generation Method For All-Hexahedral Meshing. *14th International Meshing Roundtable*, pages 377–398, 2005.
16. T.J. Tautges, T.D. Blacker, and S.A. Mitchell. The Whisker Weaving algorithm: A Connectivity-based Method for Constructing All-hexahedral Finite Element Meshes. *International Journal for Numerical Methods in Engineering*, 39:3327–3349, 1996.
17. T.J. Tautges and S.E. Knoop. Topology Modification of Hexahderal Meshes using Atomic Dual-Based operations. *12th International Meshing Roundtable*, pages 415–423, 2003.
18. T.J. Tautges and S.A. Mitchell. Whisker Weaving: Invalid Connectivity resolutions and Primal Construction Algorithm. *4th International Meshing Roundtable*, pages 115–127, 1995.