Identifying combinations of tetrahedra into hexahedra: a vertex based strategy

Jeanne Pellerin\textsuperscript{a,}* , Amaury Johnen\textsuperscript{a} , Jean-Francois Remacle\textsuperscript{a}

\textsuperscript{a}Université catholique de Louvain, iMMC, Avenue Georges Lemaitre 4, bte L4.05.02, 1348 Louvain-la-Neuve, Belgium

Abstract

Indirect hex-dominant meshing methods rely on the detection of adjacent tetrahedra that may be combined to form hexahedra. In this paper we introduce an algorithm that performs this identification and builds the set $H$ of all possible combinations of tetrahedral elements of an input mesh $T$ into hexahedra. All identified hexahedral elements are valid for engineering analysis. The new method first computes all combinations of eight vertices whose connectivity in $T$ matches the connectivity of a hexahedron. The subset of tetrahedra of $T$ triangulating each potential hexahedron is then determined. Quality checks allow to early discard poor quality hexahedra and to dramatically improve the efficiency of the method. Each potential hexahedron is computed only once. Around 3 millions potential hexahedra are computed in 10 seconds on a laptop. We finally demonstrate that the set of potential hexes $H$ built by our algorithm is significantly larger than those built using predefined patterns of subdivision of a hexahedron in tetrahedral elements.

$©$ 2017 The Authors. Published by Elsevier Ltd.
Peer-review under responsibility of the organizing committee of IMR 26.

Keywords: Hexahedral element; Hex-dominant mesh; Indirect meshing; Hexahedron decomposition; Tetrahedron combination

1. Introduction

Hexahedral meshes are considered by most of the finite element practitioners to be superior to tetrahedral meshes (see e.g. \cite{1}). Yet, no robust hexahedral meshing technique is able to process general 3D domains and generating hexahedral meshes in an automatic manner is still considered as the ultimate goal in mesh generation \cite{2}. Recently, promising techniques producing meshes composed of a majority of hexahedra have been proposed \cite{3,4,5,6}. The five steps of these indirect hex-dominant meshing methods can be summarized as follows:

1. A set of mesh vertices $V$ is initially sampled in the domain.
2. A tetrahedral mesh $T$ is built by connecting $V$, e.g. using a Delaunay kernel like \cite{8}.
3. A set $H$ of potential hexahedra that can be constructed by combining some tetrahedra of $T$ is created.
4. A maximal subset $H_c \subset H$ of compatible hexahedra is determined. It has been shown that this stage can be formally written as a maximal clique problem \cite{6}.
5. The tetrahedra, $T'$, that are not combined into hexahedra are combined into prisms, pyramids, or remain unchanged in the final hex-dominant mesh.

\* Corresponding author. Tel.: +32 10472355; fax: +32 10472999.
E-mail address: Jeanne.Pellerin@uclouvain.be

1877-7058 © 2017 The Authors. Published by Elsevier Ltd.
Peer-review under responsibility of the organizing committee of IMR 26.
In this paper we focus on the third step of this workflow and propose a new algorithm to find a set $H$ that is actually the largest possible.

In 2D, the only way to triangulate a quadrilateral without adding any vertex is to split it into two triangles. However, assuming that all potential quadrilaterals can be obtained by combining pairs of triangles sharing an edge is incorrect. Figure 1 gives a configuration in which the perfect matching algorithm [9] does not permit to build the all-quads mesh of the best quality. In 3D, the challenge is that there are at least ten different decompositions of a hexahedron into five, six, or seven tetrahedra. To compute $H$, two main approaches have been proposed. The first relies on a predefined set of patterns of the decomposition of a hexahedron into tetrahedra [3,6,7], the second on patterns of edge connections in a hexahedron [4,5]. Their main limitation is that they do not build the largest set of potential hexahedra $H$. For example, they do not detect the two hexahedra of Figure 2 (see also §2.4).

In this paper, we introduce an algorithm that detects all possible combinations of tetrahedra into hexahedra. The algorithm is based on the local search of combinations of eight vertices that are adequately connected to build a hexahedron. The key advantages of the new algorithm are that it computes all possible potential hexahedra, computes each of them once only, discards bad quality hexes at an early stage, is easy to implement and is very efficient.

After reviewing the main methods to combine tetrahedra into hexahedra (§2), we detail our vertex-based algorithm to identify the set $H$ of potential hexahedra in a tetrahedral mesh (§3). We finally demonstrate that the set of potential hexahedra $H$ built by our algorithm is larger than those built by existing methods (§4). The C++ code implementing the methods of this paper is open-source and will be available in Gmsh (www.gmsh.info).
2. Background

Before giving details on subdivisions of hexahedra into tetrahedra (§2.2) and on existing methods to identify such subdivisions in a tetrahedral mesh $T$ (§2.3), we define the terms and notations used throughout the paper (§2.1).

2.1. Definitions

Hexahedra considered in meshing are actually hexahedral finite elements. They have six quadrilateral faces, eight vertices and twelve edges. Their topology is the one of a cuboid, however their faces may not be planar and the defined volume may be non convex (Figure 3). Both the topology and the geometry of a hexahedron are unambiguously defined by its eight vertices. Their order defines the hexahedron topology (edges and faces) and their coordinates define its geometry. In this paper, we use the following adjacency templates: the six faces of a hexahedron $abcdefgh$ are $abcd$, $efgh$, $abfe$, $dcgh$, $bcgf$, $adhe$, and its twelve edges are $ab$, $bc$, $cd$, $da$, $ae$, $bf$, $cg$, $dh$, $ef$, $fg$, $gh$, $eh$ (Figure 4.1).

When a hexahedron is decomposed into tetrahedra, its quadrilateral faces are divided into two triangles. With the 12 hexahedron edges, the 6 face diagonals constitutes the set of boundary edges: $e_{bd}$ of the decomposition (grey on Figure 4.2). The remaining edges are the interior edges: $e_{int}$ (black on Figure 4.2). They either link opposite vertices of the hexahedron (e.g. $df$), or opposite vertices in a face (e.g. $eb$ and $cf$). The tetrahedra of the decomposition, denoted $t$, may be separated into two subsets: the boundary tetrahedra, denoted $t_{bd}$, whose four vertices are in the same hexahedron facet (Figure 4.4), and the interior tetrahedra, denoted $t_{int}$ (Figure 4.3). In previous works [3,5–7], boundary tetrahedra are called slivers. We do not use that term which refers to a geometrical property (degeneracy) of tetrahedra.

All identified potential hexahedra should be suitable for engineering analysis and we discard from $H$ all hexes that are invalid, i.e. that have a negative Jacobian determinant. To guarantee that the Jacobian determinant is strictly positive, we use the efficient implementation [10] of the robust and exact test proposed by Johnen et al. [11].

---

Figure 3: Hexahedral elements are not perfect cubes: (1) their faces may be not planar, (2) their volume could be concave.

Figure 4: Hexahedron topology used throughout the paper and definition of the elements of the decomposition of a hexahedron into tetrahedra.
2.2. Decomposing a hexahedron into tetrahedral elements

If theoretical results are available for the subdivision of a 3-cube into tetrahedral elements [12], little is known in the more general case of the hexahedral element.

Tetrahedrizations of the 3-cube [12]. The 3-cube has exactly 74 tetrahedrizations:

1. Every tetrahedrization of the 3-cube contains either a regular tetrahedron (i.e. a tetrahedron whose 6 edges are of equal lengths) or a diameter, i.e. an interior edge joining two opposite vertices (red edges on Figure 5).
2. There are 2 tetrahedrizations with a regular tetrahedron, symmetric to one another. The tetrahedrizations containing an interior edge are completely classified modulo symmetries by their dual complex which can be one of the last five shown on Figure 5. There are respectively 8, 24, 12, 24, 4 tetrahedrizations in each class.

A dual complex (Figure 5) is a practical way to visualize the 6 different possible decompositions (tetrahedrizations) of the 3-cube. In the dual complex, also called dual graph, one vertex corresponds to one tetrahedron and two vertices are connected by an edge if the corresponding tetrahedra are adjacent through a triangular facet. A 2-cell of the dual complex (cycle in the dual graph) corresponds to an interior edge of the tetrahedrization. In a meshing context, these different possible decompositions of the 3-cube were identified by Meshkat and Talmor [3] who enumerate the feasible dual complex graphs, called RF-graph in their paper.

Tetrahedrization of an almost perfect cube. Recently, the work of Meshkat and Talmor [3] was extended by Botella et al. [6] and Sokolov et al. [7] who proposed four additional decomposition patterns into seven tetrahedra (Figure 6). The hexahedron is split into two prisms by a tetrahedron without any facet on the hexahedron boundary and containing two interior edges. For this tetrahedron to have a strictly positive volume, it is sufficient to move slightly one of its vertices. Note that these configurations are quite common when a Delaunay kernel is used to create the tetrahedral mesh of a point set.

Bounds on the number of tetrahedra. The Euler characteristic gives a relationship between the number of tetrahedra $N_t$ in a hexahedron decomposition and its number of interior edges $N_{int}$. Each decomposition is a 3-ball with Euler characteristic $\chi = 1$, where $\chi = N_v - N_e + N_f - N_t$. Then $N_v - N_{e_{int}} - N_{e_{bd}} + N_{f_{int}} + N_{f_{bd}} - N_t = 1$. We have $N_v = 8$ vertices, $N_{f_{bd}} = 6 \times 2$ triangular boundary faces and $N_{e_{int}} = 12 + 6$ boundary edges. Since there are 4 triangular faces per tetrahedron and 2 tetrahedra per interior triangular face, we have $4N_t = 2N_{f_{int}} + N_{f_{bd}}$, then $N_{f_{int}} = 2N_t - 6$.

The number of tetrahedra $N_t$ in a hexahedron decomposition without internal points depends only on the number of interior edges: $N_t = 5 + N_{e_{int}}$. Since there are at most $\binom{8}{3} - N_{e_{bd}} = 28 - 18 = 10$ interior edges, we have the trivial bounds $5 \leq N_t \leq 15$. See also Edelsbrunner et al. [13] for additional combinatorial results.
2.3. Combining tetrahedra into hexahedra: state of the art

To compute the set $H$ of potential hexahedra that may be built by combining the elements of a tetrahedral mesh $T$ without modifying its connectivity there are two known approaches. Meshkat and Talmor\cite{3} propose to find combinations of tetrahedra into hexahedra by searching the adjacency graph of $T$ for all occurrences of the cube decomposition dual complexes (Figure 5). The problem of matching subgraphs in large sparse graphs is solved using standard data mining algorithms that operate on graphs. The same technique is used by Levy and Liu\cite{14}, Botella et al.\cite{6} and Sokolov et al.\cite{7} who additionally consider four decompositions into seven tetrahedra (Figure 6).

The second approach proposed by Yamakawa and Shimada\cite{4} relies on the vertices and edges of the tetrahedral mesh $T$. Local searches are performed into the vertex-edge graph of $T$ using two patterns. These vertex connectivity patterns generalize those proposed by Meshkat and Talmor\cite{3} and relax partially the dependency on the tetrahedral mesh. This method has been implemented by Baudouin et al.\cite{5} where a third pattern taking into account configurations with an interior flat tetrahedron was added.

Some other approaches like H-Morph \cite{15} combine tetrahedra into hexahedra, while allowing for modifications of the connectivity and geometry of the input tetrahedral mesh (tetrahedron flips, node insertions, and node displacement). This great flexibility can make the algorithm untractable, but one advantage is that it maintains a valid mixed mesh throughout the procedure.

2.4. Motivations for a new approach

Important observations led us to work on improving these existing techniques. First, they do not identify the largest set $H$ of potential hexahedra. On Figure 2 we gave two valid hexahedra that would neither be found by Meshkat and Talmor\cite{3}’s method nor by Yamakawa and Shimada\cite{4}’s method. The matching decomposition graphs are given in Figure 7. The first is a decomposition that encompasses one internal vertex, a configuration that may occur when a Steiner point is added when generating the tetrahedra. The second is a decomposition that has 8 interior tetrahedra. It is a counter example to Sokolov et al.\cite{7}’s claim that there is no hexahedron decomposition with more than 7 interior tetrahedra. Both decompositions are not identified when searching for hexahedron made of 5, 6, or 7 interior tetrahedra. They are not identified by Yamakawa and Shimada\cite{4}’s algorithm since none of their constitutive tetrahedra has three facets on the hexahedron boundary.

Second, as mentioned by Yamakawa and Shimada\cite{4}, several hexahedra may be defined using the same decomposition pattern by modifying the ordering of the vertices (Figure 8). The hexahedra have different edges and different faces while having the same tetrahedral decomposition. The hexahedron on the left being a perfect cube, the one on the right is undoubtedly invalid (zero Jacobian determinant), but were the vertices in a more general position, both could be valid.
Third, the existing methods identify the same hexahedron several times. That number is as high as the number of corner tetrahedra in the decomposition Yamakawa and Shimada\cite{4}'s approach and depends on dual complex symmetries in Meshkat and Talmor\cite{3}'s approach.

With those observations in mind, we believe that an algorithm that finds all potential hexahedra in a tetrahedral mesh should not be based on a predefined set of patterns.

### 3. A new algorithm that computes all potential hexahedra

We detail in this section our algorithm to detect combinations of tetrahedra into hexahedra and find the largest set $H$ of potential hexahedra for a given input tetrahedral mesh $T$. Sets of eight vertices that are adequately connected to build a hexahedron are first identified (§3.1). Then the tetrahedra that triangulate each potential hexahedron are subsequently determined (§3.2).

#### 3.1. Computation of the vertices of potential hexahedra

Eight vertices of the tetrahedral mesh $T$ define a potential hexahedron if (1) the twelve hexahedron edges are edges of $T$ and if (2) the six quadrilateral hexahedron faces can be formed by merging two triangular facets of $T$. This starting point is quite general and allows to automatically detect potential hexahedra without having to define a priori decomposition patterns into tetrahedra. The tetrahedral decompositions are computed in a second step allowing to discover these patterns including “exotic” ones (Figure 2).

Let us see now how to find eight vertices that are good candidates to be the corners of one hexahedron (Figure 9). Start from vertex $a$ of the tetrahedral mesh. Then, identify three different vertices $b$, $d$, and $e$ that are adjacent to $a$ through edges of $T$. Four other vertices have to be found to complete the hexahedron. Vertex $c$ should be adjacent to vertices $b$ and $d$, vertex $f$ should be adjacent to vertices $b$ and $e$, vertex $h$ should be adjacent to vertices $d$ and $e$, and finally vertex $g$ should be adjacent to vertices $c$, $f$, and $h$ (Figure 9). The existence of each of the 6 quadrilateral faces of the potential hexahedron $abcdefgh$ is verified when its four vertices are available. Face $abcd$, for example, is valid if either faces $abd$ and $bdc$, or faces $acb$ and $adc$ exist in the tetrahedral mesh. Note that checking the existence of one of the face diagonal edges $ac$ or $db$ is not sufficient to ensure the existence of the triangular facets. It is further ensured that the existing triangular faces do belong to the same parts of the input model, or model faces. Two modifications dramatically accelerate the procedure:
Potential hexahedra are only created once. Applying the procedure described previously to each vertex of the mesh leads to detect 24 times the same potential hexahedron $h \in H$, 3 for each of the 8 vertices. To avoid this repetition, we use a global order on the vertices of $T$ and apply the following restrictions when searching for adjacent vertices: $v_b > v_a; v_d, v_e > v_b; v_c, v_f, v_g, v_h > v_a$. These simple constraints allow to find every configuration once and only once.

Bad quality hexahedra are discarded early. The quality $q_{\text{quad}}$ of a quadrilateral face corner $(v_1, v_2, v_3)$ is given by $q_{\text{quad}} = \sin(\hat{v_1v_2}, \hat{v_1v_3})$. The quality $q_{\text{hex}}$ of a hexahedron corner $(v_1, v_2, v_3, v_4)$, is bounded by the quality of the 3 incident quadrilateral face corners and can be computed through a box product of the three unit vectors:

$$q_{\text{hex}} = \frac{|(\hat{v_1v_2} \times \hat{v_1v_3}) \cdot \hat{v_1v_4}|}{||\hat{v_1v_2}|| ||\hat{v_1v_3}|| ||\hat{v_1v_4}||}$$

This quality measure is referred to as the scaled Jacobian [16] and its minimal value at the 8 corners gives an upper bound on the quality of the hexahedron under construction. A quality bound can then be computed when a vertex is added. If this bound becomes smaller than the required minimum quality, the hexahedron construction is stopped.

The attentive reader may have recognized that our algorithm is a procedure to compute all possible combinations of eight vertices of $T$ fulfilling connectivity constraints. These constraints depend only on the type of cells under construction and can be implemented to identify potential prisms or pyramids. There are several choices to implement a combination generation algorithm. One can use a very general recursive form (Algorithm 1) or simple some nested for loops. For efficiency and maintenance reasons, our implementation is based on 8 nested for loops.

3.2. Computation of the tetrahedra inside a potential hexahedron

Once the vertices of a potential hexahedron $h$ are determined, we need to identify the tetrahedra of $T$ that are inside $h$. Starting from a tetrahedron $t$ that is inside $h$ we recursively add to the decomposition all the tetrahedra adjacent to $t$ that are inside hexahedron $h$. A tetrahedron $t’$ adjacent to $t$ is inside $h$ either if the facet they share does not have its three vertices in the same face of $h$ or if the four vertices of $t’$ are in the same facet of $h$ (boundary tetrahedron). The starting interior tetrahedron $t_0$ is either a tetrahedron whose four vertices are vertices of the hexahedron, or a
Loop on all the vertices of $T$;

Function main:

foreach vertex $v$ in $T$ do
    current_cell_vertices ← $v$;
    compute_cell_vertices($T$, current_cell_vertices, 1);
end

Recursive computation of all vertex combinations matching the cell template $P$;

Function compute_cell_vertices($T$, current_cell_vertices, $i$):

if $i == \text{nb_vertices}(P)$ then save_cell(current_cell_vertices);
else
    foreach $v \in \text{candidate_vertices}(T, P, \text{current_cell_vertices}, i)$ do
        if !check_cell_vertex_validity($P$, $v$, current_cell_vertices, $i$) then continue;
        current_cell_vertices ← current_cell_vertices $\cup$ $v$;
        compute_cell_vertices($T$, current_cell_vertices, $i + 1$);
    end
end

Algorithm 1: Vertex based search algorithm of potential cells $P$ (hexahedra, prisms, pyramids) in a tetrahedral mesh $T$. check_cell_vertex_validity checks if a set of selected vertices can be completed into a valid cell, candidate_vertices returns all vertices to consider to complete an input vertex set.

tetrahedron that has a face on the hexahedron boundary as well as a positive volume. There may be several valid decompositions of $h$ into tetrahedra depending on the number of boundary tetrahedra included. A simple strategy is to compute the decomposition that includes all possible existing boundary tetrahedra. All other valid decompositions may be deduced from that one by removing boundary tetrahedra.

4. Results

We have applied our algorithm to 12 different tetrahedral meshes\footnote{Input meshes can be downloaded at: \url{www.hextreme.eu}}. They were generated using the point placement strategy described in Baudouin et al.\cite{5} and implemented in Gmsh (\url{www.gmsh.info}). They have between 127 and more than 3 million vertices.

The results of our algorithm number of detected potential hexahedra and computational times for all datasets are given in Table 1. The number of potential hexahedra mainly depends on the number of vertices of the input tetrahedral mesh and on the minimal required quality (scaled Jacobian). As expected, for a given input mesh, the higher the minimal quality, the faster the algorithm. For example, the running time on dataset Knuckle decrease from 275s to less than 8s when quality increases (Table 1). The discard of hexahedra with too small scaled Jacobians is key for the efficiency of the algorithm. The multi-threaded version of our algorithm is very fast with about 300,000 potential hexahedra built per second on a laptop with 16Go RAM and an Intel(R) Core(TM) i7-6700HQ CPU @ 2.60 GHz processor. The running time clearly depends almost only on the number of potential hexahedra detected.

Comparison with pattern based methods. We compare the number of potential hexahedra identified by our algorithm with the number of potential hexahedra identified by pattern-matching methods\cite{6,7} or node combination\cite{4} in Table 2. To count the potential hexahedra matching one of the six cube decompositions or one of the four decompositions into 7 tetrahedra we use the dual complex graphs. To count the potential hexahedra that would be detected by Yamakawa and Shimada\cite{4}, we count those containing a tetrahedron that has three facets on the hexahedron facets. On small models, our algorithm detects 4 to 5% more potential hexahedra than the existing methods. That number does depend on the input tetrahedral mesh and on the minimal quality required. On larger meshes, the small difference between methods can be explained by the point placement strategy of the input tetrahedral mesh. The points are generated by propagation from the boundary of the model. Where the fronts collide, a roughly 2-dimensional surface,
Table 1: Number of valid potential hexahedra identified in 12 tetrahedral meshes with our algorithm. For each dataset are given: the number of vertices, the running times and number of hexahedra identified for minimal scaled Jacobian values between 0 and 0.9. Running times depend on the input mesh number of vertices and on the minimal value on the scaled Jacobian. They are obtained on a laptop with 16Go RAM and a Intel(R) Core(TM) i7-6700HQ CPU @2.60 GHz processor. Input tetrahedral meshes are available at: www.hextreme.eu.

| Qmin | Cube | # hex | Fusee | Crankshaft | Fusee,1 | Crankshaft,2 | Caliper | Fusee,2 | Crankshaft,2 | Fusee,3 | Lose1 | Knuckle |
|------|------|-------|-------|------------|---------|-------------|---------|--------|------------|---------|-------|---------|
|      | 0    | 127   | 11975 | 23,245     | 31,947  | 140,985     | 161,888 | 221,780| 370,401    | 501,021 | 583,561| 3,058,481|
|      | 0.1  | 0.039 | 0.756 | 1.625      | 7.415   | 13.84       | 14.468  | 8.219 | 13.645     | 51.204  | 63.815| 79.512    |
|      | 0.2  | 0.022 | 0.597 | 0.959      | 1.290   | 10.597      | 12.487  | 6.745 | 11.316     | 42.492  | 50.312| 179.264   |
|      | 0.3  | 0.002 | 0.507 | 3.913      | 1.970   | 9.631       | 12.848  | 5.043 | 13.004     | 37.994  | 50.312| 179.264   |
|      | 0.4  | 0.002 | 0.507 | 3.913      | 1.970   | 9.631       | 12.848  | 5.043 | 13.004     | 37.994  | 50.312| 179.264   |
|      | 0.5  | 0.001 | 0.507 | 3.913      | 1.970   | 9.631       | 12.848  | 5.043 | 13.004     | 37.994  | 50.312| 179.264   |
|      | 0.6  | 0.001 | 0.507 | 3.913      | 1.970   | 9.631       | 12.848  | 5.043 | 13.004     | 37.994  | 50.312| 179.264   |
|      | 0.7  | 0.001 | 0.507 | 3.913      | 1.970   | 9.631       | 12.848  | 5.043 | 13.004     | 37.994  | 50.312| 179.264   |
|      | 0.8  | 0.001 | 0.507 | 3.913      | 1.970   | 9.631       | 12.848  | 5.043 | 13.004     | 37.994  | 50.312| 179.264   |
|      | 0.9  | 0.001 | 0.507 | 3.913      | 1.970   | 9.631       | 12.848  | 5.043 | 13.004     | 37.994  | 50.312| 179.264   |

Table 2: Percentage of the potential hexahedra detected by our method that correspond to cube decompositions, to decompositions identified by [6,7], or by [4]. Minimum quality was set at 0.3. The lower the quality of the tetrahedral mesh for combination purposes the higher the gap between the number of hexahedra detected by existing methods and ours.

| Model   | #vertices | # hex | Cube | FT47 | FT47 adapt. |
|---------|-----------|-------|------|------|-------------|
| Cube    | 127       | 326   | 94.47| 97.54| 97.07       |
| Fusee   | 11,975    | 73,776| 82.67| 95.62| 97.44       |
| CrankShaft | 23,245    | 145,528| 81.16| 95.14| 97.36       |
| Fusee,1 | 71,947    | 879,028| 90.64| 99.46| 99.64       |
| Caliper | 130,572   | 1,839,192| 90.6 | 99.31| 99.54       |
| crankshaft2 | 140,985   | 2,410,584| 90.6 | 99.31| 99.54       |
| Fusee,2 | 161,888   | 2,543,913| 90.6 | 99.31| 99.54       |
| crankshaft2 | 161,888   | 2,543,913| 90.6 | 99.31| 99.54       |
| FT47    | 370,401   | 8,419,295| 87.42| 99.65| 99.77       |
| FT47 adapt. | 221,780   | 5,043,456| 87.42| 99.65| 99.77       |
| Lose1   | 583,561   | 13,582,190| 87.42| 99.65| 99.77       |
| Knuckle | 3,058,481 | 43,161,531| 87.42| 99.65| 99.77       |
point placement is not optimal. It is in this area that our method makes a difference, and the bigger the mesh is, the relatively smaller this area. Note that the higher the required minimum scaled Jacobian, the smaller the difference between the number of potential hexahedra detected by our method and the number of hexahedra detected by the existing methods. This is no surprise since the best quality is obtained for hexahedra that are close to the perfect cube which has a limited number of decompositions.

**Examples of hex-dominant meshes (Figure 10).** To demonstrate that hex-dominant meshes may be generated from potential hexahedra identified by our algorithm, we implemented two additional functionalities. The first is a simple greedy algorithm that selects the subset $H_c \subset H$ of compatible hexahedra that will be part of the output mesh. The mutually compatible hexahedra of $H_c$ are computed by adding iteratively hexahedra of decreasing quality. The compatibility of two hexahedral elements is tested by the second functionality. Two hexahedra of $H$ are compatible if their intersection is either empty or an element of their boundary: a shared vertex, a shared edge, or a shared quadrilateral face. If two hexahedra $h_i, h_j \in H$ share one interior tetrahedron, they are not compatible. If the two hexahedra share one, two, or four vertices, then we check that the corresponding vertex, edge, or face is indeed on the boundary of both hexahedra. If the two hexahedra share three vertices, they cannot be compatible.

The resulting hex-dominant meshes for three of the input tetrahedral meshes are shown on Figure 10. These meshes are the result of a greedy selection of the best quality potential hexahedra generated by our algorithm.

5. Conclusion

We presented an efficient algorithm to find the set $H$ of potential hexahedra that can be built by combining elements of a given tetrahedral mesh (1) it identifies all the valid hexahedral elements since no assumption is made on subdivisions of a hexahedron into tetrahedra, (2) each hexahedron is detected only once, (3) bad quality hexahedra are discarded early, and (4) the algorithm is easy to implement. The C++ code is open-source and will be available in Gmsh (www.gmsh.info). Contrary to previous works, our algorithm does not depend on a predefined set of templates, it is possible to recover hexahedra that have internal vertices and decompositions into eight or more interior tetrahedra (Figure 2). We have also shown that using a predefined set of templates does not permit to detect all the potential hexahedra in a given tetrahedral mesh. The percentage of the missed potential hexahedra may be significant and reach 5% when quality requirements are low.

In this paper we addressed one of the steps of the indirect hex-dominant meshing workflow. The new algorithm solves the combinatorial problem of the identification, in a given tetrahedral mesh, of all the possible combinations of tetrahedral elements into hexahedra. This algorithm is to be part of a complete workflow to build hex-dominant meshes of which all steps have a crucial impact on the final output. The quality and number of potential hexahedra generated by our algorithm highly depend on the input tetrahedral mesh $T$. The output of our algorithm is a set of potential hexahedra $H$ among which the elements of the final mesh $H_c$ are to be selected. To produce examples of hex-dominant meshes we used a greedy algorithm to perform this selection. However, this is by no means the optimal solution, i.e. a maximal clique of the compatibility graph of all potential hexahedra. Ongoing investigations show that meshes containing more hexahedra can be obtained in a reasonable amount of time by iteratively optimizing subgraphs of the solution.

**Acknowledgments**

This research is supported by the European Research Council (project HEXTREME, ERC-2015-AdG-694020).
Figure 10: Hexahedral dominant meshes generated by greedy selection of the best quality hexahedra among those identified by our algorithm (white: hexahedra, red: tetrahedra). All potential hexahedra are identified in typically less than a minute, the greedy selection runs in a few seconds.

References

[1] S. E. Benzley, E. Perry, K. Merkley, B. Clark, G. Sjaardema, A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis, in: In Proceedings, 4th International Meshing Roundtable, 1995, pp. 179–191.
[2] J. F. Shepherd, C. R. Johnson, Hexahedral mesh generation constraints, Engineering with Computers 24 (2008) 195–213.
[3] S. Meshkat, D. Talmor, Generating a mixed mesh of hexahedra, pentahedra and tetrahedra from an underlying tetrahedral mesh, International Journal for Numerical Methods in Engineering 49 (2000) 17–30.
[4] S. Yamakawa, K. Shimada, Fully-automated hex-dominant mesh generation with directionality control via packing rectangular solid cells, International journal for numerical methods in engineering 57 (2003) 2099–2129.
[5] T. C. Baudouin, J.-F. Remacle, E. Marchandise, F. Henrotte, C. Geuzaine, A frontal approach to hex-dominant mesh generation, Advanced Modeling and Simulation in Engineering Sciences 1 (2014) 1.
[6] A. Botella, B. Levy, G. Caumon, Indirect unstructured hex-dominant mesh generation using tetrahedra recombination, Computational Geo-
Early results show promising results for the proposed method, with potential applications in various fields. Further research is needed to fully evaluate the method's effectiveness and limitations. In conclusion, the proposed approach offers a new perspective for mesh generation, which could be valuable in both theoretical and practical applications.

References:
[1] J. Pellerin et al., Procedia Engineering 00 (2017) 000–000.

[7] D. Sokolov, N. Ray, L. Untereiner, B. Levy, Hexahedral-Dominant Meshing, ACM Transactions on Graphics 35 (2016) 1–23.
[8] H. Si, TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator, ACM Transactions on Mathematical Software 41 (2015) 1–36.
[9] J.-F. Remacle, J. Lambrechts, B. Seny, E. Marchandise, A. Johnen, C. Geuzainet, Blossom-Quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm, International Journal for Numerical Methods in Engineering 89 (2012) 1102–1119.
[10] A. Johnen, J.-C. Weil, J.-F. Remacle, Robust and Efficient Validation of Linear Hexahedral Elements, in: 26th International Meshing Roundtable, Barcelona, Spain, 2017.
[11] A. Johnen, J.-F. Remacle, C. Geuzaine, Geometrical validity of curvilinear finite elements, Journal of Computational Physics 233 (2013) 359–372.
[12] J. A. De Loera, J. Rambau, F. Santos, Triangulations: structures for algorithms and applications, number v. 25 in Algorithms and computation in mathematics, Springer, Berlin ; New York, 2010.
[13] H. Edelsbrunner, F. Preparata, D. West, Tetrahedrizing point sets in three dimensions, Journal of Symbolic Computation 10 (1990) 335–347.
[14] B. Levy, Y. Liu, L. p. Centroidal Voronoi Tessellation and its applications, ACM Transactions on Graphics 29 (2010) 1.
[15] S. J. Owen, S. Saigal, H-Morph: an indirect approach to advancing front hex meshing, International Journal for Numerical Methods in Engineering 49 (2000) 289–312.
[16] P. M. Knupp, Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities, Part II A framework for volume mesh optimization and the condition number of the Jacobian matrix, International Journal for Numerical Methods in Engineering 48 (2000) 1165–1185.