

COMBINATORIAL METHODS IN GRID BASED MESHING

Henrik Stromberg¹

Valentin Mayer-Eichberger²

Armin Lohrengel³

¹*Technische Universität Clausthal, Germany henrik@askdrq.com*

²*Universität Potsdam, Germany valentin@mayer-eichberger.de*

³*Technische Universität Clausthal, Germany lohrengel@imw.tu-clausthal.de*

ABSTRACT

This paper describes a novel method of generating hex-dominant meshes using pre-computed optimal subdivisions of the unit cube in a grid-based approach. Our method addresses geometries that are standard in mechanical engineering and often must comply with the restrictions of subtractive manufacturability. A central component of our method is the set of subdivisions we pre-compute with Answer Set Programming. Despite being computationally expensive, we obtain optimal meshes of up to 35 nodes available to our method in a template fashion. The first step in our grid-based method generates a coarse Precursor Mesh for meshing complete parts representing the bar stock. Then, the resulting mesh is generated in a subtractive manner by inserting and fitting the pre-generated subdivisions into the Precursor Mesh. This step guarantees that the elements are of good quality. In the final stage, the mesh nodes are mapped to geometric entities of the target geometry to get an exact match. We demonstrate our method with multiple examples showing the strength of this approach.

Keywords: mesh generation, Answer-Set-Programming, computational geometry, hex-dominant

1. INTRODUCTION

Meshes containing elements of high quality are of utmost importance for mechanical engineering applications using Finite Element Method (FEM). Current industry solutions require huge amount of manual geometry preprocessing and mesher configuration for FEM solvers to produce accurate solutions with reasonable computation resources. Our work addresses this issue by proposing a grid-based meshing approach that produces good quality hex-dominant meshes that also allow the element types prisms, pyramids and tetrahedra. Our novel method uses multiple algorithmic techniques such as heuristic and combinatorial methods. This paper explains our method, highlights the central concepts and demonstrates our method with examples.

One key step of our algorithm uses the decompositions of a coarser grid into subdivisions of the elements

that are intersected by the target geometry. The combinatorial explosion of this step is overcome by pre-computing all possible subdivisions using Answer Set Programming (ASP). We are able to compute optimal solutions for all required cases.

The paper is structured as follows. In Section 2 we define our terminology used throughout the paper and discuss mesh and element quality. Then in Section 3 we discuss how our work relates to the state-of-art meshing algorithms. In Section 4 we describe our core method and in Section 5 we explain the generation of subdivisions and present statistics of their characteristic. Section 6 and Section 7 go into detail of the main steps of the algorithm. More examples are analysed and evaluated in Section 8. Finally, we conclude and discuss future work in Section 9.

2. BACKGROUND

We choose the following terminology throughout this paper. The terms distinguish between entities of the mesh, its dual graph and the constraining geometry:

Term	Definition
Node	Mesh node
Graph node	Node of a Graph
Edge	Mesh edge
Graph edge	edge of a graph
Face	Element face in the mesh
Element	Element in the mesh
Corner	Geometry node
Curve	Geometry edge (may be curved)
NURBS	Geometry surface
Body	Geometry volume enclosed by NURBS
Precursor Mesh	Mesh from which the part can be made by subtraction
Retrenched Mesh	Mesh after cutting of all nodes out of the target geometry
Super Element	Element with twice the intended edge length to be decomposed into smaller sub elements later in the pipeline

Our approach computes hex-dominant meshes with the application of FEM. For structural applications, hex-dominant meshes can yield more accurate simulation results as opposed to fully hexahedral meshes as their elements are of higher quality (see e.g. [1]). Fully hexahedral meshes often exhibit low quality elements in regions with high stresses, which reduces the overall solution quality.

Our work addresses quality of mesh elements for FEM and we discuss established quality metrics. Finite Elements must be convex and inversion free to be usable in this context. This property can be proved by showing that the Scaled Jacobian of all elements is positive [2]. Besides this requirement a wide range of mesh quality metrics exist. Furthermore, generated meshes shall be conformal and must not exhibit hanging nodes or edges.

3. RELATED WORK

This paper relates to multiple fields in the meshing community.

Several approaches are known for the generation of quad meshes in FEM. While octree based mesh generators can directly create volume meshes, Delaunay triangulation based meshers or whisker weaving based algorithms first create a surface mesh and then a volume mesh starting from the surface. Whisker weaving

algorithms directly create hexahedral meshes.

The hexahedral meshes created through octree approaches cannot be used for FEM, if they contain hanging nodes and result in non-conformal meshes. They can be removed by using so called octant cutout templates, which decompose each hexahedral octant in multiple tetrahedrons, thereby eliminating all hanging nodes. This was an inspiration for our approach working on hybrid meshes. Their resulting mesh is tetrahedral [3].

There are also octree based mesh generators which use all hexahedral octants such as by Borden et. al. [4]. However, implementations, which are available for productive use such as Snappyhexmesh, do not include techniques like the ones shown by [4].

Delaunay based algorithms first create a triangulation and then try to generate hexahedrons from there (e.g. [5]). Whisker weaving algorithms first create a volume mesh, based on a quadrilateral surface mesh, without defining node locations, thus only generating the mesh topology. Then node locations are set and faulty elements are resolved. The algorithms work well on simple box-like geometries, but struggles with more complex parts [6].

A field of active research is in algorithms that generate structured meshes. There are no unique definitions of structured meshes across the literature. In the context of our work, we use the definition of [7] and evaluate the structure of a mesh by examining its singularity graph or generally the valence of mesh nodes. Thus, instead of a precise definition, the topic can be conquered with traits expected from a structured mesh. These traits are: low maximum valency (number of edges at a given node) over the whole mesh, a grid like structure and conservation of part symmetries.

The common way of acquiring structured meshes are sweeping algorithms. They create a volume mesh by sweeping a planar mesh through the volume (see e.g. [8]). Sweeping mesh generators yield meshes with very good structural properties but are very limited in terms of accepted geometry. A common use case is to have the user dissect the geometry into sweepable sections. However, sweepability of the complete geometry is not guaranteed by all sections being sweepable and may also depend on the meshing order of the sections. This approach is widely used in commercial software such as ANSYS [9]. It requires experienced users and often causes unexpected results.

A current development are quasi-structured quadrilateral meshing algorithms such as [10] which relax the stated requirements for structured meshes and then can achieve an automated tool chain.

There are publications describing subtractive operations on meshes such as [4] or [11]. They perform a cutting operation on a given mesh by first altering node positions to have a closed loop of edges on the

intersection curve between the initial mesh and the cutting tool. Then they remove elements and try to improve the resulting mesh.

Even though this approach was a starting point for the work presented here, it proved unusable to solve the problem of cutting a complex geometry out of a mesh. The main issue is that it does not guarantee to preserve part edges. A second issue is that the known method can create elements with critically low Scaled Jacobians. [11] shows an example where a chamfer is applied to a hex nut. The minimum Scaled Jacobian is low. Other publications such as [12] on subtractive meshing techniques actually perform a remeshing thus leaving the scope of the article.

Grid-based meshing algorithms such as the method described by Owen and Shelton in [13] use a grid of cubes which envelopes the desired geometry. Then boundary surfaces of the geometry are mapped onto the mesh. Finally, all sections of the mesh, which are outside of the target geometry, are cut from the mesh. Such methods yield good results on parts with high similarity of the generated grid to the target geometry but produce poor quality meshes on parts with deviating geometry. Their results are also sensitive to the orientation of the target geometry in global coordinates [7].

Livesu et al. in [14] proposes a similar grid-based method using Super Elements and refined subdivisions. However, their work considers only hexahedral elements. The computational problem to find subdivision becomes easier in their case and their method cannot guarantee minimal quality of the resulting elements.

Many approaches with similar properties to the ones discussed exist, but follow the same general concepts. All known implementations of meshing algorithms use a Boundary REPresentation (BREP) geometry interface neglecting Computer Aided Design (CAD) tree structure. Our method takes advantage of the expression from the CAD tree and to the best of our knowledge this has not been before in the context of automatic hex-dominant meshing.

Our study of subdivisions of the cube is related to other combinatorial approaches investigating similar problems: In [15] the computation of all possible decompositions of a single hexahedron into tetrahedrons is done through Boolean satisfiability solving, which use similar algorithmic techniques than ASP. Their method is limited to the 8 nodes of the hexahedron itself and introduces no further interior nodes. In a related combinatorial approach, the decompositions of pyramids into hexahedra is studied in [16]. The use of Integer Programming in meshing is demonstrated in [17] to optimize a balanced octree in a grid-based

meshing algorithm.

4. ALGORITHM

Our meshing algorithm consist of four stages:

1. Generate Precursor Mesh
2. Assign Super Elements
3. Map Mesh to Geometry
4. Optimize Mesh

A brief description of the steps is as follows: The first stage chooses a coarse Precursor Mesh from the target geometry as a starting point for the pipeline. Then, to each element in the Precursor Mesh a Super Element is assigned to roughly match the geometry restrictions. In the third step each node in the finer mesh of the Super Elements is mapped back to the target geometry. Finally, the mesh is optimized to improve its quality.

To generate conformal meshes, the algorithm requires a complete collection of Super Element decompositions for any subset of the 8 corner nodes of the unit cube. This collection constitutes a main contribution of our work and the generation is explained in depth in Section 5. In the rest of this section we discuss the prerequisites for the steps of our algorithm and our reasoning behind the proposed method.

The costly computation for Super Elements does not scale to larger meshes with more than 35 nodes. Due to the size requirements of real-world applications of up to 10^8 nodes, we have chosen an approach inspired by subtractive manufacturing. Many geometries meshed for simulations represent mechanical parts. Such parts are typically designed to be manufactured by removing material from bar stock such as a I-beam. As the bar stock material is a manufactured in a continuous rolling process, its geometry is easily meshable with sweeping methods. From this swept mesh the material has to be removed to carve the target geometry out of the bar stock, as shown in Figure 1. The process stages are illustrated with a cube with a central cylindrical hole subtracted from it.

Just removing elements from the mesh in order to emulate the cutting of material is insufficient as doing so may create a rough surface which does not match the desired part contour. We have modeled the cutting of material by replacing all elements of the Precursor Mesh with pre-computed Super Elements. All Super Elements we use are supposed to replace elements of the Precursor Mesh thereby increasing its granularity. Each Super Element is specified by removing some of the nodes from the base element geometry. For an

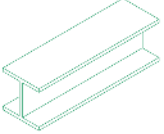
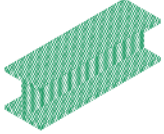
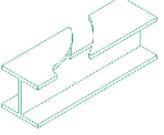
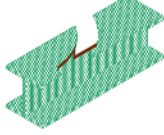
	manufacturing	meshing
sweepable production stage		
finished part		

Figure 1: Stages of part production typically for subtractive manufacturing in mechanical engineering.

eight node hexahedron $2^8 = 256$ possible Super Elements exist. In order to assure compatibility between the used Super Elements each node of the Precursor Mesh must be kept or removed for all adjacent elements and Super Elements for them must be chosen accordingly. Furthermore, we have globally defined the surface mesh of a Super Element face depending on the nodes present in the Super Element (see Figure 2). In this approach the assembly of a part from Super Elements can be seen as selecting which nodes of the Precursor Mesh are supposed to be present in the final mesh. We call the result of this process *Retrenched Mesh*. The Retrenched Mesh for the used example is shown in Figure 3 and the details of the process are described in Section 6. The node coordinates of the nodes within all Super Elements are transformed such that the outer nodes of the Super Element coincide with the corresponding nodes of the Precursor Mesh.

The resulting mesh is a rough approximation of the desired geometry. In order to make the surfaces match, all faces of the surface mesh are bound to entities of the target geometry as shown for the example in Figure 3. The details of the process are described in Section 7. Geometric entities, which are not fully represented, are deemed too small to be part of the mesh and are neglected as a consequence.

In a last step the mesh quality is improved by first optimizing the surface mesh and then the volume mesh. The surface mesh is optimized by moving surface nodes on their respective geometric entities. Nodes which are bound to corners cannot be moved. Currently elements with low Scaled Jacobian (SJ) are identified. Then the node locations of these elements are optimized one element at a time. The resulting changes are minuscule in the provided example.

The novel processes Super Element Assignment and Mesh Mapping can be implemented in $\mathcal{O}(n \log n)$ in the number of elements as shown in the respective sec-

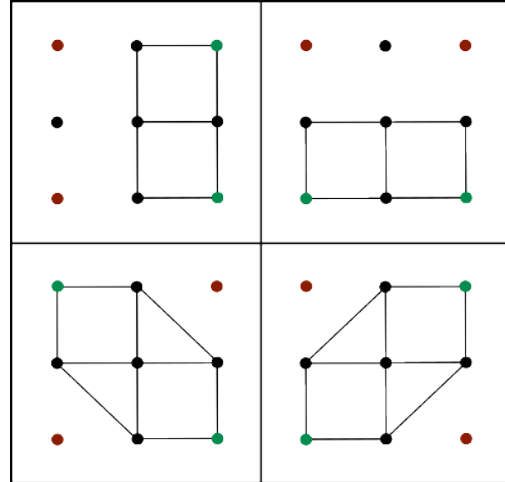


Figure 2: Example of interfaces between Super Elements. Green nodes are retained Precursor Mesh nodes, dropped nodes are marked red. Possible additional nodes be they used or unused are drawn black.

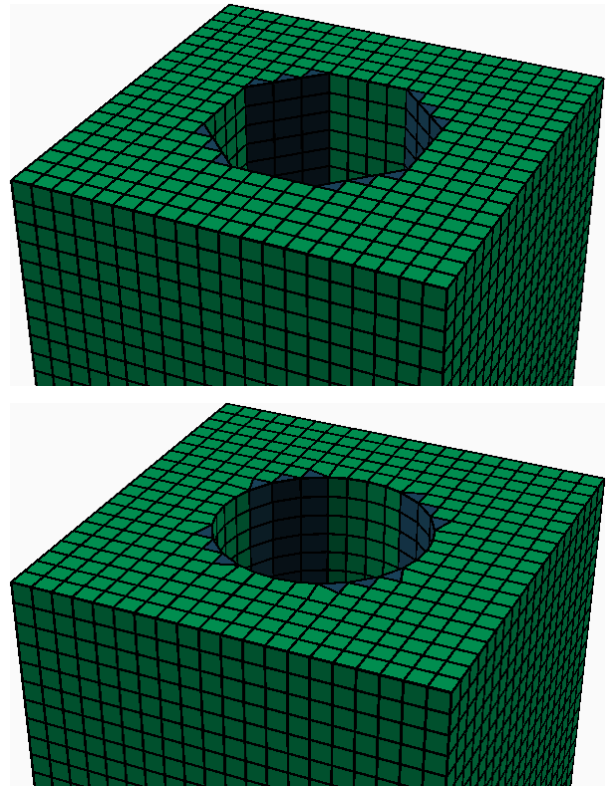


Figure 3: Top: Mesh of Example 1 after assigning Super Elements. Bottom: Mesh after being mapped to geometry. This resulting mesh was automatically computed from target geometry with our proposed method and did not require manual improvements of the mesh.

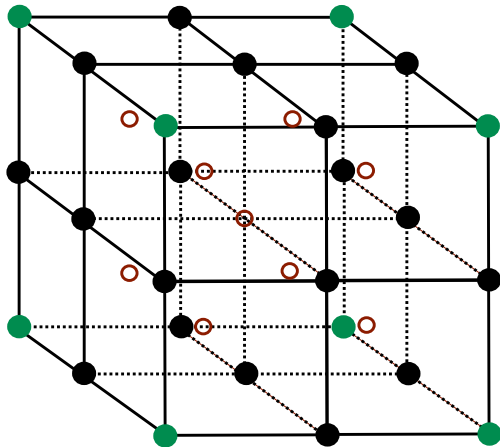


Figure 4: The set of all additional node positions for a hexahedron that are available for the generation of the subdivision. We consider all possible elements that can be generated from subsets of these nodes (Table 1).

tions. With this complexity of the complete algorithm we expect our approach to scale to typical industrial applications in simulations with up to 10^8 nodes.

5. SUPER ELEMENT GENERATION

In this section we explain the application of Answer Set Programming to generate optimal Super Elements from subdivisions of the cube.

To generate the more fine grained internal mesh of the hexahedral Super Elements, we chose additional 27 nodes located proportionally in the mid sections of the cube, in total being 35 node locations. The additional nodes provide a refinement of the Super Element. The node locations are shown in Figure 4. The main task is to compute the optimal hybrid mesh that can be created from all these nodes. A naive brute-force algorithm would not be able to solve this computational task with realistic resources and more advanced methods are necessary.

With these node locations all possible tetrahedra, hexahedra, prisms and pyramids are generated and their Scaled Jacobian is computed according to [18]. We have specified a minimum SJ for each element type in order to exclude elements of poor quality and to speed up the solution progress. Ultimately, the element portfolio listed in table 1 is used to assemble mesh solutions. Note that the SJ values for different element types are incomparable and we selected different minimal thresholds.

Table 1: Element portfolio

Element type	All elem. (SJ > 0)	Min. SJ	Count
Tetrahedrons	44850	0.16	22750
Hexahedrons	16333575	0.3	3809
Prisms	1351685	0.45	2751
Pyramids	264501	0.35	1626

We have modeled a logically consistent mesh in terms of its dual graph, enforcing the topological constraints with the following rule set:

- Each face of an element must connect to exactly one other element or be part of the outer hull.
- Any two neighboring faces on the outer hull are locally convex.
- All elements are connected (there are no disconnected sub-graphs).
- No edge of any face may intersect any other face, except for shared nodes (the mesh is not self-intersecting).

All these constraints were modelled in ASP, which is well suited for describing such graph-based constraints, and the program is natural and human readable. In particular, the connectivity constraint is difficult to formulate in related approaches such as Integer Programming or Boolean Satisfiability. We use the state-of-the-art ASP solver Clingo¹ by the Potassco group [19] to compute solutions to our problem.

There are 256 instances for which meshes have to be generated in order to get a complete set of Super Elements. Only 22 problems (plus the trivial empty mesh problem) have to be solved because many problems can be transformed into each other by rotating or mirroring the instance. Table 1 lists the number of elements with positive SJ for each type. Allowing all these elements in the optimization would generate too big instances. So, we carefully determined a minimal threshold for each element type to keep the problem instances manageable.

Our implementation generates meshes for all cases in about one minute per problem with the element portfolio from Table 1 on 8 cores with 3.6 GHz. The solver proved optimality for all solutions.

We are interested in various quality metrics of the Super Elements beyond maximizing the minimal SJ. To get a better idea if other criteria are better suited, we ran separate optimizations for the following goals:

¹<https://potassco.org/clingo/>

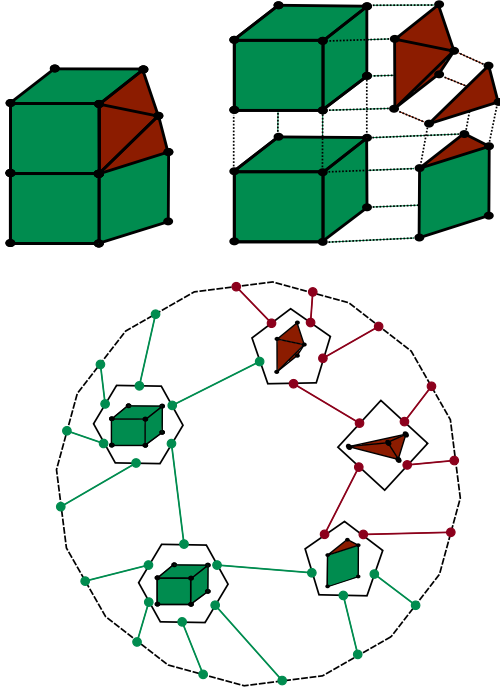


Figure 5: Graph representation of a mesh with triangular faces colored red and quadrangular faces colored green.

Table 2: Optimization results for all subdivisions with best results for the three different criteria, respectively. Usually, the three resulting meshes per case are different. A comprehensive list of for the column Min SJ according to [18] is shown in Figure 6. SJs for mixed meshes cannot be compared to hexahedral meshes.

Case	# Inst.	Max. Valence	Min SJ	Max Elem. Count
1	8	3	0.35	1
2	12	4	0.46	2
3	12	8	0.26	20
4	24	9	0.21	18
5	6	5	1.00	4
6	4	6	0.24	6
7	24	13	0.21	36
8	12	12	0.21	41
9	8	8	0.29	14
10	8	13	0.25	36
11	12	13	0.21	41
12	24	12	0.21	36
13	24	9	0.21	28
14	6	6	0.46	8
15	24	12	0.35	29
16	12	6	0.46	8
17	2	22	0.35	52
18	8	19	0.35	46
19	12	16	0.35	40
20	4	18	0.35	50
21	8	12	0.35	29
22	1	6	1.00	8

- maximize the minimum SJ
- minimize element count
- minimize the maximum node valency

For optimizing the minimum SJ we list all 20 non-trivial Super Elements in Figure 6. For most of these solutions, it would be very hard to find these by hand, and impossible to prove that they are optimal. Table 2 lists the optimization results for all 22 solved problems with values regarding the different goals. In the more complex instances, different goals lead to different meshes.

6. SUPER ELEMENT ASSIGNMENT

For each element of the Precursor Mesh a Super Element has to be assigned. Compatibility between the selected Super Elements is guaranteed by deciding which nodes on the Precursor Mesh shall be present in the final mesh and selecting the resulting Super Element based on the present nodes. So the presence of each Precursor Mesh node in the final mesh can be described as a vector of Boolean variables N .

For inserting each Super Element into each element of the Precursor Mesh a fit quality is computed by approximating the residual volume R . Fig. 7 illustrates this procedure simplified to a planar problem. This is achieved by evaluating if a set of integration points (see X in fig. 7) is inside or outside the geometry and if they are inside or outside of the Super Element. Now, an assignment of all N is searched which minimizes the sum of all R for the selected Super Elements in their respective Precursor Mesh elements. The integration points can be tested for being inside the Super Elements once as the Super Elements are not problem dependent. We are using a grid of $5 \times 5 \times 5$ integration points as this is the smallest uniform grid which is able to differentiate all 256 Super Elements.

Computing R as the sum of all deviating integration points leads to wavy mesh surfaces for plain geometries. We have found solutions to have a much higher quality when computing R as $R = R_S^2 + R_P$ where R_S is the sum of all integration points only occurring in the Super Element and R_P are integration points only occurring in the geometry. By doing so cutting more material is favoured and the algorithm becomes more stable.

We solve the minimization problem of $R(N)$ with the ASP solver Clingo to compute optimal solutions. We were able to solve the benchmark problem in Figure 3 for up to 64 elements in the Precursor Mesh with proven optimality. For larger meshes no optimum was found in reasonable computation time and the non-optimal results were not satisfactory.

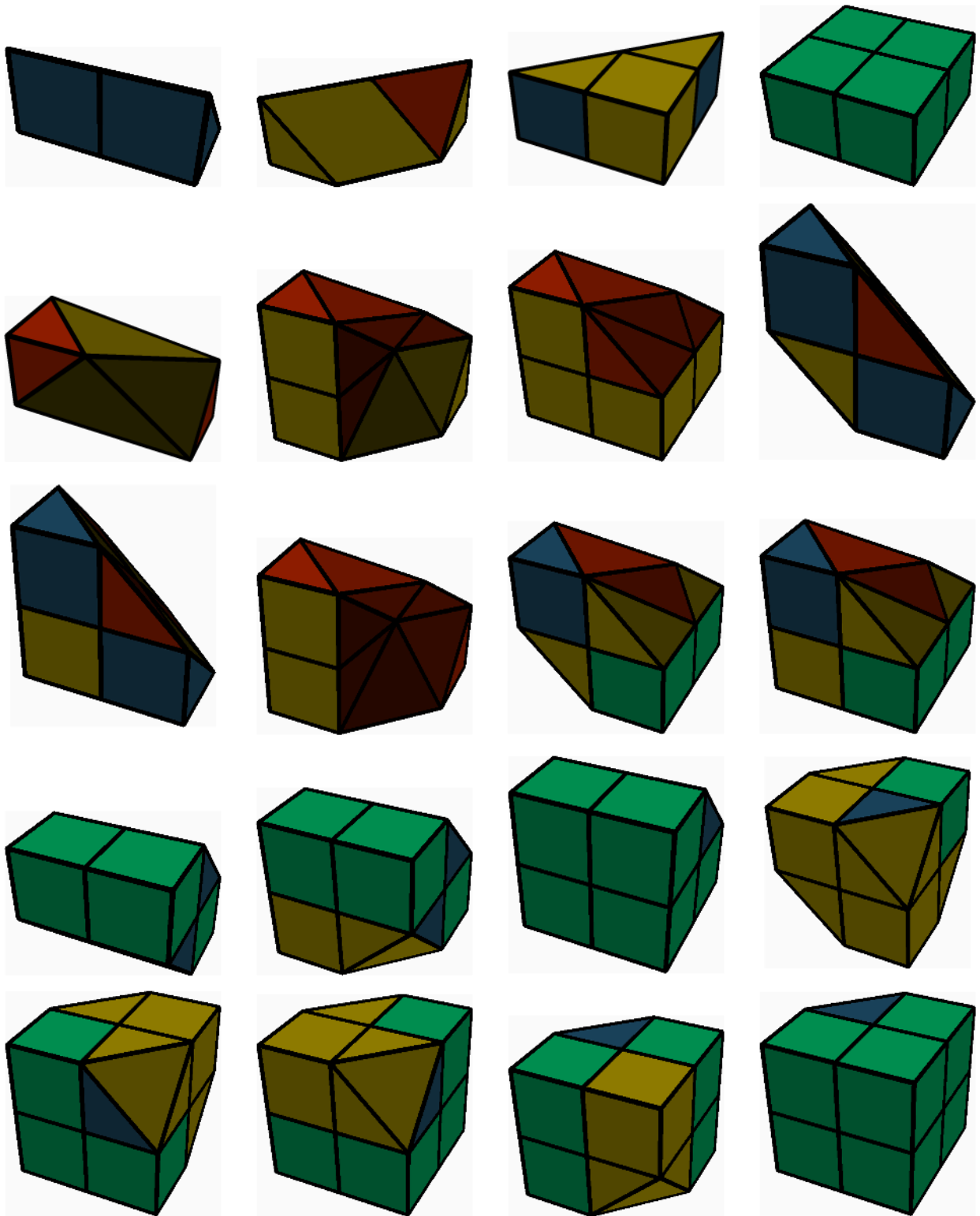


Figure 6: All non-trivial subdivisions of a unit cube maximising the minimal SJ. The element types are indicated by the following color coding: Hexahedra in green, Tetrahedra in red, Prisms in blue and Pyramids in yellow.

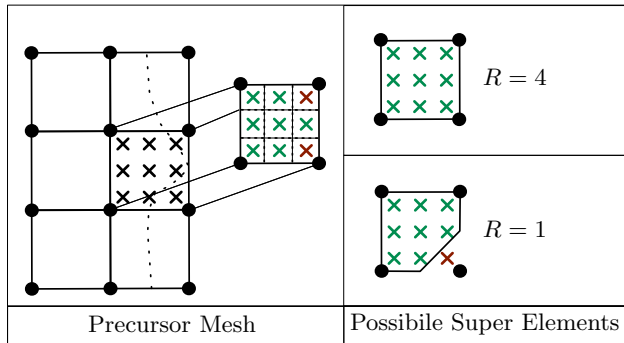


Figure 7: Assembling a mesh by assigning Super Elements to Precursor Mesh elements

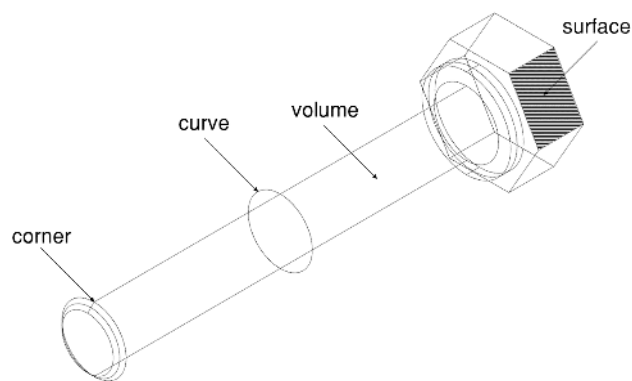


Figure 8: Geometric entities

In order to scale the overall method to real world instances we have developed a heuristic that computes good approximations for $R(N)$. This heuristic works by iterating over all elements of the Precursor Mesh, finding the most suitable Super Element for it. All elements vote whether they want adjacent nodes included or excluded from the mesh. The execution time of this algorithm is linear in element count.

When comparing the ASP based exact method to the heuristics, the two methods compute the same results on our set of sample instances. This indicates that our heuristic provides decent solutions for larger instances.

7. ENTITY MAPPING

In the entity mapping phase of the algorithm, all surface nodes are assigned to geometric entities. This allows to improve geometric accuracy, elevate mesh order and insert more fine grained elements. Geometric entities are surfaces which enclose and define volumes. They are enclosed by curves which are terminated with corners as is shown in Figure 8.

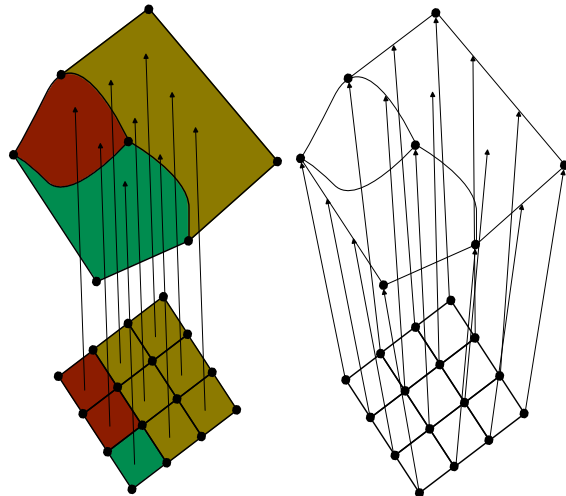


Figure 9: Entity Mapping process

For each node of the surface mesh one geometric entity has to be assigned. We achieve this assignment by first assigning a surface to each element face of the surface mesh. The surface is determined by projecting a line from the element face center in normal direction. The surface assigned to the element face is the first one intersecting this line regardless of the line parameter sign of the intersection. In this process we obtain a colored mesh (see fig. 9 left). We execute the intersection test with a fine triangular mesh of the target geometry because line intersection test with NURBS geometry are complex and unreliable. The intersection point of a NURBS surface and a line is a nonlinear problem whereas the intersection problem of a line and a surface is a linear problem. The unreliability stems from the fact that there is no way to securely test that an arbitrary nonlinear function has no roots. The triangular mesh is generated with $\frac{1}{4}$ edge length of the target mesh. When all element faces which are adjacent to a node have the same color it is assigned to this surface. In case the adjacent element faces have two different colors the node is assigned to the curve splitting the two surfaces. If the two surfaces share multiple curves the closet curve to the node is chosen. When more then two colors occur, the node must be assigned to a corner. If there is a corner with the same colors in neighbouring surfaces it is selected. Otherwise the distance between node and potentially assigned corners is used as a tiebreaker.

In this process, any geometry smaller than the element size of the mesh is automatically de-featured.

A trivial implementation of the algorithm would require to project a test line from each face of the surface mesh to each triangle of the underlying tessellation. In the worst case, the count of surface mesh faces is the

same order of magnitude as the volume element count and the total node count. In this case, the time complexity of the algorithm with respect to the total mesh nodes n is in $\mathcal{O}(n^2)$.

This run-time can be improved to $\mathcal{O}(n \log n)$ by employing a more sophisticated ray casting method such as lazy sweep [20].

8. RESULTS ON EXAMPLES

The method is tested with a subset of the MAMBO [21] data set. The Precursor Mesh for the geometries is selected automatically by analysing the CAD tree structure of the model. The results are presented in Table 3. The resulting meshes and their quality reflect the underlying model structure of the geometry. B5 is modeled as a revolution whereas B14 is modeled as an extrusion. In the case of B11 and B14 the circular cross section is detected and a specialized surface mesher is used for the 2D mesh before sweeping. The elliptic cross section of B15 is not recognized resulting in a lower quality mesh. The difference between B10 and our own example from fig. 3 is caused by model structure as well. The MAMBO example is modeled as a single extrusion and our own example is modeled as an extruded cube and a subtracted cylinder.

In general our method show good performance on locally convex geometries. For cases such as B2, which contain sharp non convex features, the method fails to generate an adequate mesh. This issue is caused by the fact that the algorithm currently only uses convex Super Elements. The results can be improved by including non-convex Super Elements.

9. CONCLUSION AND FUTURE WORK

We have presented a novel approach for grid-based meshing using heuristic and combinatorial methods. For two steps of our pipeline we applied Answer Set Programming to solve combinatorial sub-problem. The set of schemes for subdivisions of the cube may find application in other hybrid meshing algorithms. Our success of using ASP in the context of meshing may inspire other combinatorial analysis of related problems.

Our method advocates to make use of the underlying CAD model of the geometry to be meshed. Especially in the context of FEM of mechanical engineering, we believe that this focus should receive more attention. Future algorithms could compare their solutions to ours using the same CAD tree from the MAMBO geometry benchmark. These CAD models, the ASP models we have used and all the computed Super Elements are made available on Github: <https://github.com/HenrikJStromberg/>

combinatorial_meshing

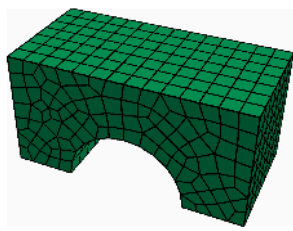
As next steps in our work, we see the following improvements. The current results of our mesh optimization seem to be inferior to those presented by [13]. We will combine their mesh optimization with our algorithm in the future. We plan to evaluate our method on more complex instances available in the MAMBO set. Finally, we aim to improve the efficiency of our implementation to tackle problem sizes common in industrial applications.

10. ACKNOWLEDGEMENTS

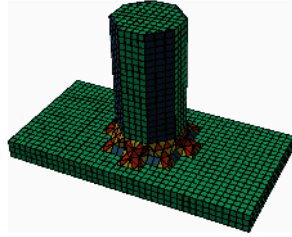
We would like to thank Jeff Erickson and Jean Christoph Jung for discussing our ideas with us and giving valuable feedback.

References

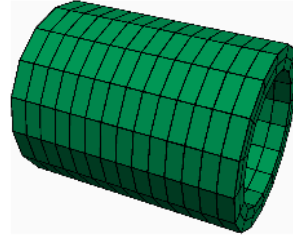
- [1] Veyhl C., Belova I., Murch G., Öchsner A., Fiedler T. “On the mesh dependence of non-linear mechanical finite element analysis.” *Finite Elements in Analysis and Design*, vol. 46, no. 5, 371–378, May 2010
- [2] Shepherd J.F. *Topological and geometric constraint-based hexahedral mesh generation*. PhD Thesis, University of Utah, USA, 2007
- [3] Yerry M.A., Shephard M.S. “Automatic three-dimensional mesh generation by the modified-octree technique.” *International Journal for Numerical Methods in Engineering*, vol. 20, no. 11, 1965–1990, 1984
- [4] Borden M.J., Shepherd J.F., Benzley S.E. “Mesh cutting: Fitting simple all-hexahedral meshes to complex geometries.” *Proceedings, 8th international society of grid generation conference*. 2002
- [5] Remacle J., Henrotte F., Baudouin T.C., Geuzaine C., Béchet E., Mouton T., Marchandise E. “A Frontal Delaunay Quad Mesh Generator Using the L^∞ Norm.” W.R. Quadros, editor, *Proceedings of the 20th International Meshing Roundtable, IMR 2011, October 23-26, 2011, Paris, France*, pp. 455–472. Springer, 2011
- [6] Tautges T.J., Blacker T.D., Mitchell S.A. “The Whisker-Weaving Algorithm: A connectivity-based method for constructing all-hexahedral finite element meshes.” *International Journal for Numerical Methods in Engineering*, vol. 39, no. 19, 3327–3349, Oct. 1996. Publisher: Wiley
- [7] Pietroni N., Campen M., Sheffer A., Cherchi G., Bommes D., Gao X., Scateni R., Ledoux F., Remacle J., Livesu M. “Hex-mesh generation



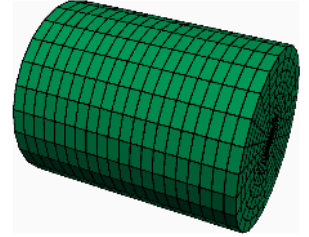
B0



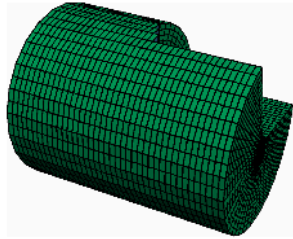
B2



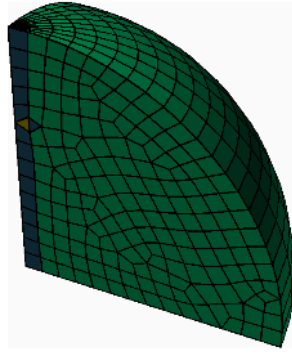
B4



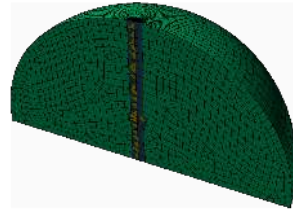
B5



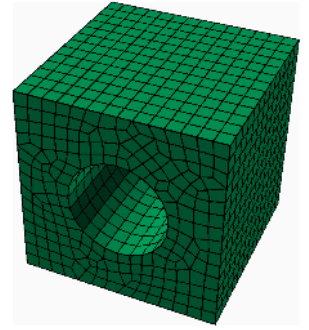
B6



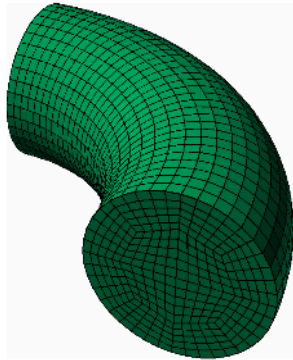
B7



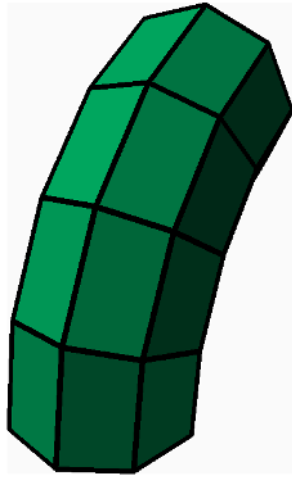
B9



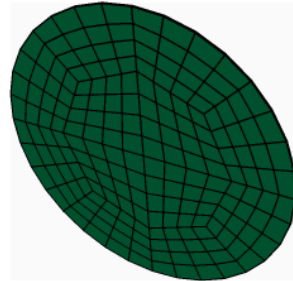
B10



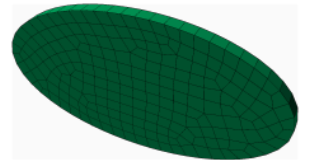
B11



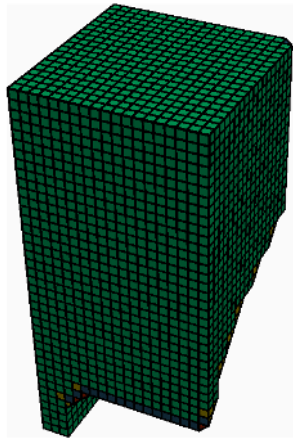
B12



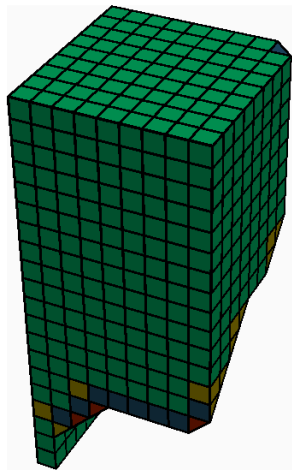
B14



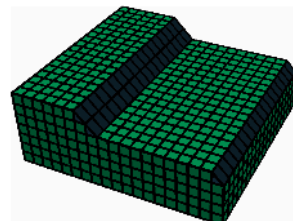
B15



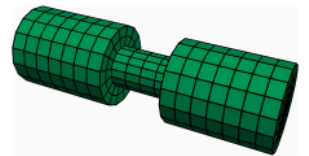
B17



B19



B21



B42

Table 3: The hex-dominant meshes automatically computed by our method from basic instances of the MAMBO geometry benchmark.

- and processing: a survey.” *ACM transactions on graphics*, vol. 42, no. 2, 1–44, 2022
- [8] Jankovich S.R., Benzley S.E., Shepherd J.F., Mitchell S.A. “The Graft Tool: An All-Hexahedral Transition Algorithm for Creating a Multi-Directional Swept Volume Mesh.” K. Shimada, editor, *Proceedings of the 8th International Meshing Roundtable, South Lake Tahoe, California, USA, October 10-13, 1999*, pp. 387–392. 1999
- [9] Ansys Inc. “Ansys Mechanical 2022 R2.”, 2022
- [10] Reberol M., Georgiadis C., Remacle J.F. “Quasi-structured quadrilateral meshing in Gmsh—a robust pipeline for complex CAD models.” *arXiv preprint arXiv:2103.04652*, 2021
- [11] Zhu H., Gao S., Xian C. “Hexahedral Mesh Cutting Using Geometric Model With New Boundaries Well Matched.” *American Society of Mechanical Engineers Digital Collection*, pp. 147–156, Sep. 2013
- [12] Dhondt G. “A new automatic hexahedral mesher based on cutting.” *International Journal for Numerical Methods in Engineering*, vol. 50, no. 9, 2109–2126, 2001
- [13] Owen S.J., Shelton T.R. “Evaluation of grid-based hex meshes for solid mechanics.” *Engineering with Computers*, vol. 31, no. 3, 529–543, Jul. 2015
- [14] Livesu M., Pitzalis L., Cherchi G. “Optimal dual schemes for adaptive grid based hexmeshing.” *ACM Transactions on Graphics (TOG)*, vol. 41, no. 2, 1–14, 2021
- [15] Pellerin J., Verhetsel K., Remacle J.F. “There Are 174 Subdivisions of the Hexahedron into Tetrahedra.” *ACM Trans. Graph.*, vol. 37, no. 6, 2018
- [16] Verhetsel K., Pellerin J., Remacle J.F. “A 44-Element Mesh of Schneiders’ Pyramid.” X. Roca, A. Loseille, editors, *27th International Meshing Roundtable*, Lecture Notes in Computational Science and Engineering, pp. 73–87. Springer International Publishing, Cham, 2019
- [17] Pitzalis L., Livesu M., Cherchi G., Gobbetti E., Scateni R. “Generalized adaptive refinement for grid-based hexahedral meshing.” *ACM Transactions on Graphics (TOG)*, vol. 40, no. 6, 1–13, 2021
- [18] Lobos C. “Towards a unified measurement of quality for mixed-elements.” *Tech. Rep. 2015/01*, 2015
- [19] Gebser M., Kaminski R., Kaufmann B., Schaub T. “Multi-shot ASP solving with clingo.” *Theory and Practice of Logic Programming*, vol. 19, no. 1, 27–82, 2019
- [20] Silvia C.T., Mitchell J.S. “The Lazy Sweep Ray Casting Algorithm for Rendering Irregular Grids.” *IEEE transactions on visualization and computer graphics*, vol. 3, no. 2, 1997
- [21] Ledoux F. “MAMBO.”, Jun. 2022. URL <https://gitlab.com/franck.ledoux/mambo>