An agglomeration-based multilevel-topology concept with application to 3D-FE meshes

J.K. Kraus, J. Synka

RICAM-Report 2004-08
AN AGGLOMERATION-BASED MULTILEVEL-TOPOLOGY
CONCEPT WITH APPLICATION TO 3D-FE MESHES

JOHANNES K. KRAUS AND JOSEF SYNKA

ABSTRACT. In this paper we focus on generating hierarchies of topological relations
for unstructured three-dimensional (3D) finite element (FE) meshes. The related
topological elements are agglomerates, faces, edges, and vertices. We provide practi-
cal definitions and propose efficient procedures, in particular for the setup of coarse
faces and coarse edges. Moreover, we examine the agglomeration algorithm intro-
duced by Jones and Vassilevski [3] and suggest a proper generalization for 3D
meshes.

This setting allows for a recursive construction of a sequence of nested coarse
topologies, which can be used in different areas of finite element computations, e.g.,
load balancing for parallel processing, domain decomposition methods, algebraic
multigrid or multilevel preconditioning methods based on incomplete factorization.

1. INTRODUCTION

The numerical solution of partial differential equations by the method of finite ele-
ments involves large structured or unstructured meshes for discretization. Regarding
the quality of an FE approximation (difference between numerical and exact solu-
tion measured in some norm), there are many situations where a significant portion
of degrees of freedom in the FE model can be saved if the underlying mesh is well
adapted to the problem. Due to complicated shapes of the computational domain
and/or the necessity of local mesh refinement this adaptation often causes a loss of
structure in the mesh. In many FE simulations unstructured meshes are used right
from the beginning.

Now for different reasons one might like to subdivide a mesh into small(er) parts;
mainly for the efficient iterative solution of the arising system of linear algebraic
equations, techniques like load balancing for parallel processing, domain decomposi-
tion methods or algebraic multigrid are used, which strongly depend on proper mesh
partitionings, a question we want to deal with in this paper.

Usually, one demands certain properties of the partitioning:

• One desirable property is that the resulting subsets (blocks) of elements are
  strongly connected in the sense that for each pair of elements in one block
  (partition) there is a path along element faces, possibly via other elements (in
the same block), that connects the two elements to each other; (Note that two elements share a face iff the shortest connecting path has length one.)

- Another favored property of the partitioning is that all or most of the partitions contain approximately the same number of elements.

- A third common aim is to keep the number of faces shared by elements belonging to different blocks as small as possible; (This is equivalent to a low edge cut when partitioning the vertices of a dual graph. In this dual graph every element corresponds to one vertex; two of its vertices are connected via an edge if and only if the corresponding elements in the FE mesh share a face.)

We have these requirements in mind when constructing partitionings of FE meshes via element agglomeration.

Since any mesh partitioning problem defines an equivalent (constrained) graph partitioning problem (for a dual graph) various techniques from this area apply [1, 2, 4, 5, 6, 7]. A summary of the functionality of a number of publically available software packages for graph partitioning was given in [9]. Other partitioning methods for unstructured meshes based on agglomeration and aggregation techniques have been investigated in the context of algebraic multigrid [3, 10, 11, 12].

However, we will focus on a general multilevel-topology concept for 3D finite element meshes in this paper. This includes the construction of a sequence of nested agglomerated elements, referred to as agglomerates hereafter, and the setup as well as the transfer of certain topological relations between agglomerates, faces, and edges, from one level (of agglomeration) to the next. The definition of these topological elements and their relations will be presented in Section 2. Given an arbitrary partitioning of fine-level agglomerates, i.e., a coarse-agglomerate-fine-agglomerate relation, Section 3 is concerned with the construction of a coarse topology, i.e., coarse faces, coarse edges, coarse vertices, and their relations. Sections 4 and 5 deal with agglomeration algorithms: A very fast algorithm–linear complexity (in the number of faces)–has been introduced in [3]. For unstructured 2D meshes this algorithm produces well-shaped and suitably matched agglomerates. Moreover, up to boundary effects, it correctly reproduces structured triangular and quadrilateral meshes on coarser levels. We will recall this algorithm, which, in principle, is not limited to 2D applications, in Section 4. However, it will be shown that the quality of agglomerates can be improved considerably by introducing some major modifications in the 3D case. We thus propose an improved algorithm, described in Section 5, to be used for the agglomeration process (in 3D space), which our multilevel-topology concept is based on. Finally, we will discuss some results obtained for 3D unstructured (tetrahedral) meshes in Section 6.

2. Topological relations

A finite element mesh in general is described by geometry data (e.g., nodal coordinates) and topological relations (e.g., an element-node relation). This defines each single element as well as it determines the topological relationship between elements, faces, edges, and vertices. We identify the nodes of the FE mesh one-to-one with the vertices of its finest-level (level-0) topology.
In this paper, we will use topological relations only, in order to investigate agglomeration-based partitionings of a set of elements arising from an arbitrary finite element discretization. For that reason, let us consider a bounded computational domain $\Omega$ in three-dimensional space, i.e., $\bar{\Omega}$ is a compact set, and, w.l.o.g., assume that $\Omega$ has exactly one connected component. Moreover, let $T$ denote a finite element mesh, that is, a set of finite elements $T := \{ t_i : i = 1, 2, \ldots n_T \}$, the most natural assumptions for which are:

$$t_i \cap t_j = \emptyset \quad \text{for} \quad i \neq j,$$

and

$$\bigcup_{i=1}^{n_T} t_i = \bar{\Omega}.$$

Our framework is composed of the following topological elements:

**Definition 2.1.** [Level-$m$ agglomerates]. The set of level-$0$ agglomerates is defined by $A_0 := \{ a_{i_0} \equiv t_{i_0} : i_0 = 1, 2, \ldots, n_{A_0} \}$, i.e., the level-$0$ agglomerates coincide with the finite elements (per definition). Moreover, for $1 \leq m \leq l$, a set $A_m := \{ a_{i_m} : i_m \in I_{A_m} := \{ 1, 2, \ldots, n_{A_m} \} \}$ is called a set of level-$m$ agglomerates if each $a_{i_m}$ is the union of level-$(m-1)$ agglomerates $a_{j_{m-1}}$, i.e.,

$$a_{i_m} = \bigcup_{j_{m-1} \in I_{A_m}} a_{j_{m-1}},$$

such that the index sets $I_{A_m}$ provide a partitioning of $I_{A_{m-1}}$, the index set of level-$(m-1)$ agglomerates, i.e.,

$$I_{A_m} \cap I_{A_{m-1}} = \emptyset \quad \text{for} \quad i_m \neq j_{m-1},$$

and

$$\bigcup_{i_m \in I_{A_m}} I_{A_m} = I_{A_{m-1}} := \{ 1, 2, \ldots, n_{A_{m-1}} \}.$$

For the set of finite elements (level-0 agglomerates) an element-face relation, a face-edge relation, as well as an edge-vertex relation might be provided by the mesh generator. However, if not directly accessible, these relations can also be computed from the element-vertex relation based on the following definitions. Here $V$, $\partial V$, and $V \setminus \partial V$ denote the set of all vertices, the set of boundary vertices, and the set of interior vertices, respectively.

**Definition 2.2.** [Faces]. Consider a set $A = \{ a_i : i \in I_A \}$ of agglomerates. An intersection $(a_i \cap a_j) \subset V$ of a pair of different agglomerates $a_i$ and $a_j$ in $A$ is called an interior face, connecting $a_i$ and $a_j$, if there is no $k \in I_A$ such that $(a_i \cap a_j) \subset (a_i \cap a_k)$ or $(a_i \cap a_j) \subset (a_k \cap a_j)$. An intersection of the type $(a_i \cap \partial V)$ is called a boundary face of $a_i$, if it is not contained in any intersection of the type $(a_k \cap \partial V)$ or $(a_i \cap a_k)$.

The union of the set of interior faces $F \setminus \partial F$ and the set of boundary faces $\partial F$ defines the set of all faces $F$.

That means, faces (in terms of vertices) are maximal intersections of two agglomerates or of an agglomerate with the boundary. This uniquely determines an agglomerate-face and a face-vertex relation. Now, given faces in terms of vertices, one can compute maximal intersections of faces (in terms of vertices), defining a face-edge relation as well as an edge-vertex relation.
Definition 2.3. [Edges]. Consider $\mathcal{F} \setminus \partial \mathcal{F} = \{ f_i : i \in \mathcal{I}_{\mathcal{F} \setminus \partial \mathcal{F}} \}$, the set of interior faces, and $\partial \mathcal{F} = \{ f_i : i \in \mathcal{I}_{\partial \mathcal{F}} \}$, the set of boundary faces. An intersection $(f_i \cap f_j) \subset \mathcal{V}$ of a pair of different faces $f_i$ and $f_j$ in $\mathcal{F}$ is called maximal for the face $f_i$ if there is no face $f_k \in \mathcal{F}$, $k \notin \{ i, j \}$, such that $(f_i \cap f_j) \subset (f_i \cap f_k)$. A maximal intersection $(f_i \cap f_j)$ for face $f_i$ is called an interior edge if $f_i$ and $f_j$ both are interior faces and if there is no boundary face $f_k \in \partial \mathcal{F}$ such that $(f_i \cap f_j) = (f_i \cap f_k)$; otherwise, it is called a boundary edge.

The union of the set of interior edges $\mathcal{E} \setminus \partial \mathcal{E}$ and the set of boundary edges $\partial \mathcal{E}$ defines the set of all edges $\mathcal{E}$.

Remark 2.1. Note that for general 3D agglomerates an intersection of two faces that is maximal for one face not necessarily has to be maximal for the other face. However, collecting all intersections that are maximal for at least one face results in a set of edges (defined in terms of vertices).

Remark 2.2. Assuming that an element-vertex relation stems from a finite element mesh consisting of tetrahedrons, hexahedrons, or mixed polyhedrons, the computation of maximal intersections can be omitted since in this case a face will always contain at least three vertices and an edge will always contain exactly two vertices. This considerably speeds up the setup of the initial topological relations.

The topology concept composed of agglomerates, faces, edges, and vertices thus is enhanced by introducing certain basic topological relations, which (in the general case of a multilevel topology) either refer to a given level, say level $m$, or describe a relationship between the topological elements of two different levels $m_1 \neq m_2$.

Definition 2.4. [Basic topological relations]. Let

$$
\mathcal{A}_m = \{ a_{i_m} : i_m \in \mathcal{I}_{\mathcal{A}_m} := \{1, 2, \ldots n_{\mathcal{A}_m} \} \},
\mathcal{F}_m = \{ f_{i_m} : i_m \in \mathcal{I}_{\mathcal{F}_m} := \{1, 2, \ldots n_{\mathcal{F}_m} \} \},
\mathcal{E}_m = \{ e_{i_m} : i_m \in \mathcal{I}_{\mathcal{E}_m} := \{1, 2, \ldots n_{\mathcal{E}_m} \} \},
\mathcal{V}_m = \{ v_{i_m} : i_m \in \mathcal{I}_{\mathcal{V}_m} := \{1, 2, \ldots n_{\mathcal{V}_m} \} \},
$$

respectively denote the set of agglomerates, of faces, of edges, and of vertices at level $m$, $m \in \{0, 1, 2, \ldots, l\}$. Moreover, let $\mathcal{M}_m := \{ \mathcal{A}_m, \mathcal{F}_m, \mathcal{E}_m, \mathcal{V}_m \}$ be the set of topological elements (agglomerates, faces, edges, and vertices) at level $m$.

Then, for $0 \leq m_1, m_2 \leq l$ and any pair $(B_{m_1}, C_{m_2})$ of sets with $B_{m_1} \in \mathcal{M}_{m_1}$ and $C_{m_2} \in \mathcal{M}_{m_2}$, a subset $\mathcal{R}_{B_{m_1}, C_{m_2}} \subset \mathcal{I}_{B_{m_1}} \times \mathcal{I}_{C_{m_2}}$ of the Cartesian product of the index sets $\mathcal{I}_{B_{m_1}}$ and $\mathcal{I}_{C_{m_2}}$ is called a basic topological relation between levels $m_1$ and $m_2$.

Some examples for basic topological relations at a given level $m$:

- agglomerate-face: $\mathcal{R}_{\mathcal{A}_m, \mathcal{F}_m} \subset \mathcal{I}_{\mathcal{A}_m} \times \mathcal{I}_{\mathcal{F}_m}$
- face-edge: $\mathcal{R}_{\mathcal{F}_m, \mathcal{E}_m} \subset \mathcal{I}_{\mathcal{F}_m} \times \mathcal{I}_{\mathcal{E}_m}$
- edge-vertex: $\mathcal{R}_{\mathcal{E}_m, \mathcal{V}_m} \subset \mathcal{I}_{\mathcal{E}_m} \times \mathcal{I}_{\mathcal{V}_m}$

Some examples for basic topological relations between two consecutive levels:

- coarse-agglomerate-fine-agglomerate: $\mathcal{R}_{\mathcal{A}_{m-1}, \mathcal{A}_m} \subset \mathcal{I}_{\mathcal{A}_{m-1}} \times \mathcal{I}_{\mathcal{A}_m}$
- coarse-face-fine-face: $\mathcal{R}_{\mathcal{F}_{m-1}, \mathcal{F}_m} \subset \mathcal{I}_{\mathcal{F}_{m-1}} \times \mathcal{I}_{\mathcal{F}_m}$
- coarse-edge-fine-edge: $\mathcal{R}_{\mathcal{E}_{m-1}, \mathcal{E}_m} \subset \mathcal{I}_{\mathcal{E}_{m-1}} \times \mathcal{I}_{\mathcal{E}_m}$
Besides these basic topological relations we also define topological product relations that relate two sets of topological elements, which may even coincide, via a third one.

**Definition 2.5.** [Topological product relations]. For $0 \leq m_1, m_2, m_3 \leq l$, and any triple $(B_{m_1}, C_{m_2}, D_{m_3})$ of sets $B_{m_1} \in \mathcal{M}_{m_1}$, $C_{m_2} \in \mathcal{M}_{m_2}$, and $D_{m_3} \in \mathcal{M}_{m_3}$, a subset $R_{B_{m_1}, C_{m_2}, D_{m_3}} = R_{B_{m_1}, C_{m_2}} \cdot R_{C_{m_2}, D_{m_3}} \subset I_{B_{m_1}} \times I_{D_{m_3}}$ of the Cartesian product of the index sets $I_{B_{m_1}}$ and $I_{D_{m_3}}$ is called a topological product relation (between levels $m_1$ and $m_3$) if $R_{B_{m_1}, C_{m_2}}$ and $R_{C_{m_2}, D_{m_3}}$ are basic topological relations and if for any pair of indices $(i_{m_1}, k_{m_3}) \in R_{B_{m_1}, C_{m_2}}$, $D_{m_3}$ there exists (at least) one index $j_{m_2}$ such that $(i_{m_1}, j_{m_2}) \in R_{B_{m_1}, C_{m_2}}$ and $(j_{m_2}, k_{m_3}) \in R_{C_{m_2}, D_{m_3}}$.

**Remark 2.3.** In a similar manner, products of more than two basic topological relations can be defined, which yields a product of two general topological relations, i.e., each factor being either a basic or a product relation.

Some examples for topological product relations:

- agglomerate-face-edge: $R_{A_m, F_m, E_m} = R_{A_m, F_m} \cdot R_{F_m, E_m} \subset I_{A_m} \times I_{E_m}$
- agglomerate-face-edge-vertex:
  $$R_{A_m, F_m, E_m, V_m} = R_{A_m, F_m, E_m} \cdot R_{E_m, V_m} \subset I_{A_m} \times I_{V_m}$$
- face-edge-face: $R_{F_m, E_m, F_m} = R_{F_m, E_m} \cdot R_{E_m, F_m} \subset I_{F_m} \times I_{E_m}$
- edge-vertex-edge: $R_{E_m, V_m, E_m} = R_{E_m, V_m} \cdot R_{V_m, E_m} \subset I_{E_m} \times I_{E_m}$

The last two examples above define equivalence relations in the usual sense, i.e., mathematical relations being reflexive, symmetric and transitive. They will be of particular interest in the agglomeration procedures which we will discuss later (in Sections 4 and 5).

Topological relations can be represented as rectangular (Boolean) sparse matrices, as suggested in [11]. The size of the matrix storing a relation $R_{B_{m_1}, C_{m_2}} \subset I_{B_{m_1}} \times I_{C_{m_2}}$ is $n_{B_{m_1}} \times n_{C_{m_2}}$. The entry in row $i_{m_1}$ and column $j_{m_2}$ is ‘1’ if $(i_{m_1}, j_{m_2}) \in R_{B_{m_1}, C_{m_2}}$ and ‘0’ otherwise. Consequently, it is sufficient to use only two one-dimensional arrays for the storage of each relation: the column indices of all nonzero entries, ordered row-wise, are stored in the first array; the $i$th entry in the second array gives the total number of nonzero entries stored for rows $1$ to $i-1$, $i = 1, 2, \ldots, n_{B_{m_1}} + 1$.

By storing also the numerical value ‘1’ of the nonzero entries of the sparse matrices representing topological relations, the product of two topological relations can be computed by first evaluating the (numerical) sparse-matrix product and then resetting all nonzero entries to unity. This corresponds to the computation of the symbolic product of two sparse matrices that only determines the (maximal) nonzero pattern of their numerical product.

3. Coarse topology

The generation of a sequence of coarse-topologies can be based on a recursive agglomeration of elements of the original finite-element mesh. However, if the agglomeration procedure used for this purpose is based on topological relations between agglomerates, faces, edges, and possibly vertices, one has to reproduce these relations at each agglomeration level. We will first deal with this problem in here, which does
not depend on a specific agglomeration algorithm, and then, in Sections 4 and 5, examine the agglomeration process itself.

It is convenient to store agglomerates in terms of faces, faces in terms of edges, and edges in terms of vertices. Although coarse vertices are not relevant for the agglomeration procedures described in this paper, we will comment on their integration briefly.

3.1. Coarse faces. Let us first look at the coarse faces. Having computed a set of level-

\[ \mathcal{A}_m \] agglomerates \( \mathcal{A}_m \) that provides a partitioning of the set of level-(\( m - 1 \)) agglomerates \( \mathcal{A}_{m-1} \) (cf., Definition 2.1), we need to determine their faces (coarse faces), i.e., we have to construct a coarse-level agglomerate-face relation \( \mathcal{R}_{\mathcal{A}_m, \mathcal{F}_m} \). We thereby assume that all fine-level relations of interest, in particular the fine-level agglomerate-face relation \( \mathcal{R}_{\mathcal{A}_{m-1}, \mathcal{F}_{m-1}} \), are already available.

It is possible to set up a procedure for the generation of coarse faces following Definition 2.2 from Section 2. However, this has some major drawbacks: firstly, the computation of maximal intersections is rather expensive; secondly, it involves coarse vertices, which have to be determined in advance; thirdly, coarse faces depending on the choice of coarse vertices may cause degeneration. If the coarse vertices are not selected carefully in this approach, agglomerates and faces, or faces and edges, defined as sets of vertices, may become identical; agglomerates may even degenerate to edges.

That is why we prefer an alternative approach, which is based on the following intuitive definition of coarse faces:

**Definition 3.1.** [Coarse faces]. Two level-

\[ \mathcal{A}_m \] agglomerates \( a_{i_m} \) and \( a_{k_m} \) share an interior face \( f_{j_m} \in (\mathcal{F}_m \setminus \partial \mathcal{F}_m) \) (connecting \( a_{i_m} \) and \( a_{k_m} \)) if and only if there exists a pair of agglomerates \((a_{p_{m-1}}, a_{q_{m-1}}) \in \mathcal{A}_{m-1} \times \mathcal{A}_{m-1}\) such that \((i_m, p_{m-1}) \in \mathcal{R}_{\mathcal{A}_m, \mathcal{A}_{m-1}}, (k_m, q_{m-1}) \in \mathcal{R}_{\mathcal{A}_m, \mathcal{A}_{m-1}}\), where the agglomerates \( a_{p_{m-1}} \) and \( a_{q_{m-1}} \) share an interior face \( f_{r_{m-1}} \in (\mathcal{F}_{m-1} \setminus \partial \mathcal{F}_{m-1}) \), i.e., \((p_{m-1}, q_{m-1}) \in \mathcal{R}_{\mathcal{A}_{m-1}, \mathcal{F}_{m-1}, \mathcal{A}_{m-1}}\). An agglomerate \( a_{i_m} \) has a boundary face \( f_{j_m} \in \partial \mathcal{F}_m \) (being part of any boundary sector) iff there exists an agglomerate \( a_{p_{m-1}} \in \mathcal{A}_{m-1} \) such that \((i_m, p_{m-1}) \in \mathcal{R}_{\mathcal{A}_m, \mathcal{A}_{m-1}}\) and \( a_{p_{m-1}} \) has a boundary face \( f_{r_{m-1}} \in \partial \mathcal{F}_{m-1} \) (being part of the same boundary sector).

The union of the set of interior faces \( \mathcal{F}_m \setminus \partial \mathcal{F}_m \) and the set of boundary faces \( \partial \mathcal{F}_m \) defines the set \( \mathcal{F}_m = \{ f_{i_m} : i_m \in \mathcal{I}_{\mathcal{F}_m} := \{ 1, 2, \ldots, n_{\mathcal{F}_m} \} \} \) of all faces at level \( m \).

At any level \( m \), \( 0 \leq m \leq l \), the information about the partitioning of faces into boundary faces and interior faces can be stored in one additional topological relation

\[ \mathcal{R}_{\mathcal{F}_m, \partial \Omega} \subset \mathcal{I}_{\mathcal{F}_m} \times \mathcal{I}_{\partial \Omega}, \]

where \( \mathcal{I}_{\partial \Omega} := \{ 1, 2, \ldots, n_{\partial \Omega} \} \) and \((i_m, k) \in \mathcal{R}_{\mathcal{F}_m, \partial \Omega}\) if and only if the face \( f_{i_m} \) belongs to the boundary sector \( k \).

**Remark 3.1.** Note that there is at most one face shared by two different agglomerates \( a_{i_m} \) and \( a_{k_m} \). Moreover, any agglomerate \( a_{i_m} \) has at most one face being part of a specific boundary sector.

Given the topological relations \( \mathcal{R}_{\mathcal{A}_m, \mathcal{A}_{m-1}} \) and \( \mathcal{R}_{\mathcal{A}_{m-1}, \mathcal{F}_{m-1}} \), Definition 3.1 induces a coarse-face-fine-face relation \( \mathcal{R}_{\mathcal{F}_m, \mathcal{F}_{m-1}} \). This leads to an algorithm which allows for
a straightforward determination of coarse-level faces and their relation to fine-level faces:

Algorithm 3.1. [Coarse-face setup].

- **Input:** relations \( R_{A_m, A_{m-1}}, R_{A_{m-1}, F_{m-1}}, \) and \( R_{F_{m}, \partial \Omega} \).
- Perform the following steps, thereby constructing \( R_{F_m, F_{m-1}}, R_{F_m, A_m}, R_{F_m, \partial \Omega} \):
  1. Compute the relation \( R_{A_m, A_{m-1}} \cdot F_{m-1} \) as well as the transposed relation \( (R_{A_m, A_{m-1}} \cdot F_{m-1})^T \).
  2. Determine all faces \( f_{s,m-1} \) in the intersection of any two coarse-level face agglomerates \( a_{i_m} \) and \( a_{j_m} \) or in the intersection of a coarse-level agglomerate \( a_{i_m} \) with the boundary (any boundary sector) and mark them.
  3. Loop while there are still marked fine-level faces:
     - Take next marked fine-level face \( f_{s,m-1} \).
     - If \( f_{s,m-1} \) belongs to the boundary (any boundary sector), then intersect the boundary (this specific boundary sector) with the agglomerate \( a_{i_m} \) to which \( f_{s,m-1} \) belongs. All fine-level faces in this intersection will now be unmarked. They are stored for the new coarse face (update \( R_{F_m, F_{m-1}} \)), which in this case is a boundary face for the agglomerate \( a_{i_m} \) (update \( R_{F_m, A_m} \) and \( R_{F_m, \partial \Omega} \)).
     - If \( f_{s,m-1} \) does not belong to the boundary (any boundary sector), then intersect the two agglomerates \( a_{i_m}, a_{j_m} \) to which \( f_{s,m-1} \) belongs. All fine-level faces in this intersection are going to be unmarked. They are stored for the new coarse face (update \( R_{F_m, F_{m-1}} \)), which in this case is an interior face between agglomerates \( a_{i_m} \) and \( a_{j_m} \) (update \( R_{F_m, A_m} \)).
  4. Compute \( R_{A_m, F_m} := (R_{F_m, A_m})^T \).

- **Output:** relations \( R_{A_m, F_m}, R_{F_m, F_{m-1}} \) and \( R_{F_m, \partial \Omega} \).

Step (2) of this algorithm involves the relations \( R_{F_{m-1}, A_{m-1}, A_m} \) and \( R_{F_{m-1}, \partial \Omega} \) and is most easily implemented by incrementing the corresponding elements of an integer vector of length \( n_{F_{m-1}} \), which has been initialized with zeros. It is easily checked that the coarse-level agglomerate-face relation \( R_{A_m, F_m} \), as obtained from Algorithm 3.1, satisfies

\[
R_{A_m, F_m} = R_{A_m, A_{m-1}} \cdot R_{A_{m-1}, F_{m-1}} \cdot R_{F_{m-1}, F_m}
\]

where \( R_{F_{m-1}, F_m} := (R_{F_m, F_{m-1}})^T \). We will also make use of the following two (coarse-level) topological product relations relating (coarse) faces among each other: First, let the adjacency of faces at level zero be defined via the relation

\[
R_{F_0, F_0} := R_{F_0, F_0, F_0} \cdot R_{F_0, E_0} = R_{F_0, E_0} \cdot R_{E_0, F_0}.
\]

Then we call

\[
R_{F_m, F_m} := R_{F_m, F_{m-1}} \cdot R_{F_{m-1}, F_{m-1}} \cdot R_{F_{m-1}, F_m}
\]

the adjacency relation for faces, at level \( m \). Note that we do not need coarse edges for determining the adjacency of coarse faces.

The second equivalence relation, which we define for faces, determines whether two
faces belong to exactly the same agglomerate or not. At level $m$ this information is provided by the product relation

$$R_{\mathcal{F}_m, \mathcal{A}_m} := R_{\mathcal{F}_m, \mathcal{A}_m} \cdot R_{\mathcal{A}_m, \mathcal{F}_m}.$$  

3.2. Coarse edges. The main difference between the coarse-face setup and the coarse-edge setup is that boundary coarse faces always have to be stored for exactly one coarse-level agglomerate and interior coarse faces for exactly two coarse-level agglomerates whereas coarse edges, in general, have to be stored for a whole bunch of coarse faces. Besides that difference the definition and setup of coarse edges is similar to that of coarse faces:

**Definition 3.2.** [Coarse edges]. Two level-$m$ faces $f_{1m}$ and $f_{km}$ share an edge $e_{jm} \in \mathcal{E}_m$ if and only if there exists a pair of faces $(f_{p_{m-1}, q_{m-1}}) \in \mathcal{F}_{m-1} \times \mathcal{F}_{m-1}$ such that $(i_{m-1}, p_{m-1}) \in R_{\mathcal{F}_{m-1}, \mathcal{F}_{m-1}}$ and $(k_{m-1}, q_{m-1}) \in R_{\mathcal{F}_{m-1}, \mathcal{F}_{m-1}}$, where the faces $f_{p_{m-1}}$ and $f_{q_{m-1}}$ share an edge $e_{r_{m-1}} \in \mathcal{E}_{m-1}$, i.e., $(p_{m-1}, q_{m-1}) \in R_{\mathcal{F}_{m-1}, \mathcal{E}_{m-1}}$. The considered edge $e_{jm}$ is called an interior edge if all of the faces $f_{um}$ for which $(r_{m-1}, u_{m}) \in R_{\mathcal{E}_{m-1}, \mathcal{F}_{m-1}}$ share an edge $e_{jm}$ such as obtained from Algorithm 3.2 fulfills $R_{\mathcal{E}_{m}, \partial \mathcal{E}_m}$ defines the set $\mathcal{E}_m := \{ e_{jm} : i_{m} \in \mathcal{I}_{\mathcal{E}_m} := \{1, 2, \ldots, n_{\mathcal{E}_m}\} \}$ of all edges at level $m$.

Again, we introduce one more relation $R_{\mathcal{E}_m, \partial \Omega}$ to store the information whether a level-$m$ edge is on the boundary or not. However, we do not distinguish different boundary sectors for boundary edges.

This yields the computations caused by coarse-level edges to be organized as follows:

**Algorithm 3.2.** [Coarse-edge setup].

- **Input:** relations $R_{\mathcal{F}_m, \mathcal{E}_m}$, $R_{\mathcal{F}_{m-1}, \mathcal{E}_{m-1}}$, and $R_{\mathcal{E}_{m-1}, \partial \Omega}$.
- **According to Definition 3.2** one constructs the following relations:
  - $R_{\mathcal{F}_m, \mathcal{F}_{m-1}, \mathcal{E}_{m-1}} = R_{\mathcal{F}_m, \mathcal{F}_{m-1}} \cdot R_{\mathcal{F}_{m-1}, \mathcal{E}_{m-1}}$
  - $R_{\mathcal{E}_{m-1}, \mathcal{F}_{m-1}, \mathcal{E}_m} = (R_{\mathcal{F}_m, \mathcal{F}_{m-1}, \mathcal{E}_{m-1}})^T$:

  Next, mark all fine-level edges in the intersection of any two coarse-level faces and loop over marked fine-level edges computing the intersection of all coarse-level faces (in terms of fine-level edges) that share the actual fine-level edge. Construct a new coarse-level edge for each such intersection;
  - Update $R_{\mathcal{E}_m, \mathcal{E}_{m-1}}$, i.e., store the fine-level edges contained in this intersection and unmark them;
  - Update $R_{\mathcal{E}_m, \mathcal{F}_m}$, i.e., store the corresponding coarse-level faces;
  - Update $R_{\mathcal{E}_m, \partial \Omega}$, i.e., decide whether the new coarse-level edge is a boundary edge or not and store this information;
  - $R_{\mathcal{F}_m, \mathcal{E}_{m}} := (R_{\mathcal{E}_m, \mathcal{F}_{m}})^T$

- **Output:** relations $R_{\mathcal{F}_m, \mathcal{E}_m}$, $R_{\mathcal{E}_m, \mathcal{E}_{m-1}}$ and $R_{\mathcal{E}_m, \partial \Omega}$.

The coarse-level face-edge relation $R_{\mathcal{F}_m, \mathcal{E}_m}$, as obtained from Algorithm 3.2, fulfills

$$R_{\mathcal{F}_m, \mathcal{E}_m} = R_{\mathcal{F}_m, \mathcal{F}_{m-1}} \cdot R_{\mathcal{F}_{m-1}, \mathcal{E}_{m-1}} \cdot R_{\mathcal{E}_{m-1}, \mathcal{E}_m}.$$
where \( \mathcal{R}_{\varepsilon_{m-1}, \varepsilon_m} := (\mathcal{R}_{\varepsilon_m, \varepsilon_{m-1}})^T \). Similar to the equivalence relations for faces we also supply the adjacency of edges at level zero, defined via the relation

\[
\mathcal{R}_{\varepsilon_0, \varepsilon_0} := \mathcal{R}_{\varepsilon_0, \nu_0, \varepsilon_0} := \mathcal{R}_{\varepsilon_0, \nu_0, \varepsilon_0},
\]

and, accordingly,

\[
\mathcal{R}_{\varepsilon_m, \varepsilon_m} := \mathcal{R}_{\varepsilon_m, \varepsilon_{m-1}} \cdot \mathcal{R}_{\varepsilon_{m-1}, \varepsilon_{m-1}} \cdot \mathcal{R}_{\varepsilon_{m-1}, \varepsilon_m}
\]

defining the adjacency relation for edges, at level \( m \). Note that (3.7) does not explicitly depend on coarse vertices.

The second equivalence relation, which we define for edges, determines whether two edges belong to one and the same face (have some face in common) or not. At level \( m \) this information is provided by the product relation

\[
\mathcal{R}_{\varepsilon_m, \mathcal{F}_m, \varepsilon_m} := \mathcal{R}_{\varepsilon_m, \mathcal{F}_m} \cdot \mathcal{R}_{\mathcal{F}_m, \varepsilon_m}.
\]

3.3. Coarse vertices. As already mentioned the agglomeration algorithms that we will consider in the next two sections do not rely on any topological relations involving coarse vertices. However, in order to apply the presented multilevel-topology concept in the area of algebraic multigrid, the selection of a coarse grid, induced by coarse vertices, may be useful. The most natural integration of coarse vertices in the considered framework is obtained from Definition 3.2 by replacing faces with edges and edges with vertices. The same applies for the setup of the additional relations involving coarse vertices.

4. The agglomeration algorithm of Jones and Vassilevski

In this section, we will focus our attention on the agglomeration process itself, which we want to base on the topological relations developed so far. Our starting point is an algorithm introduced by Jones and Vassilevski [3]. Given a set \( \mathcal{A}_{m-1} \) of level-\((m-1)\) agglomerates this algorithm can be used to construct a set \( \mathcal{A}_m \) of level-\( m \) agglomerates. It works with the topological relations

- \( \mathcal{R}_{\mathcal{F}_{m-1}, \mathcal{F}_{m-1}} \) (cf., Equation (3.3))
- \( \mathcal{R}_{\mathcal{A}_{m-1}, \mathcal{F}_{m-1}} \)
- \( \mathcal{R}_{\mathcal{F}_{m-1}, \mathcal{A}_{m-1}} \)
- \( \mathcal{R}_{\mathcal{F}_{m-1}, \mathcal{A}_{m-1}, \mathcal{F}_{m-1}} \) (cf., Equation (3.4))

and utilizes a weight function \( w : \mathcal{F}_{m-1} \rightarrow \mathbb{Z} \) that assigns integer weights to all faces \( f_k \in \mathcal{F}_{m-1} \). However, and this slightly reduces the computational costs, it also works when operating on the (reduced) set of interior faces.\(^1\)

Algorithm 4.1. [Agglomeration (Jones and Vassilevski)].

- Initialize: Set \( w(f_k) = 0 \) for all faces \( f_k \in \mathcal{F} := \mathcal{F}_{m-1} \setminus \partial \mathcal{F}_{m-1} \); set \( n_m = 0 \);
- Global search: Find a face \( f_k \in \mathcal{F} \) with maximal weight \( w(f_k) \geq 0 \); increment \( n_m := n_m + 1 \); set \( a_{n_m} = \emptyset \);
- Collect the level-\((m-1)\) agglomerates \( a_{r_{m-1}} \in \mathcal{A} := \mathcal{A}_{m-1} \) that form \( a_{n_m} \).

\(^1\)Operating on the full set of faces is done mainly for the reason of generating a proper set of coarse vertices, a goal that is related to the coarse-grid selection problem in algebraic multigrid.
(1) Set $a_{nm} := a_n \cup a_{pm} \cup a_{qm}$ where $(k, p_{m-1})$ and $(k, q_{m-1})$ in $\mathcal{R}_{F, A}$, and set $w_{\text{max}} = w(f_k), w(f_k) = -1$;
(2) Increment $w(f_i) := w(f_i) + 1$ for all faces $f_i$ with $i$ such that $w(f_i) \neq -1$ and $(k, i) \in \mathcal{R}_{F, F}$;
(3) Increment $w(f_j) := w(f_j) + 1$ for all faces $f_j$ with $j$ such that $w(f_j) \neq -1$ and $(k, j) \in \mathcal{R}_{F, F} \cap \mathcal{R}_{F, A}$;
(4) From the neighbors of $f_k$, choose a face $f_k'$ with maximal weight $w(f_k')$;
   - If $w(f_k') \geq w_{\text{max}}$: set $k := k'$ and go to step (1);
   - If $w(f_k') < w_{\text{max}}$: the level-$m$ agglomerate $a_{nm} \in A_m$ is complete;
     set $w(f_k) = -1$ for all faces $f_k$ of level-$(m-1)$ agglomerates $a_{r_{m-1}}$ contained in $a_{nm}$; go to step Global search;

**Remark 4.1.** Efficiently implemented, the execution time of the above algorithm is linear in the number of faces. This essentially is caused by the fact that only a limited $O(1)$ number of different (integer) weights are assigned to faces during the whole process. When operating with a double linked list of faces, ordered according to their actual weights, faces can easily be removed at one position or inserted at another. If in addition one also stores and updates the position of the first face in the successive block of faces with the same weight, and one does this for all $O(1)$ different weights, the computational effort for the global search for a face with a maximal weight reduces to $O(1)$. This is the most important part since the operation count for the construction of any agglomerate will then in practice stay proportional to the number of removed faces.\(^2\)

In view of the desired properties of the resulting partitioning (we formulated them in Section 1) we noticed that Algorithm 4.1 does not guarantee the connectivity of fine-level agglomerates via fine-level faces within any coarse-level agglomerate. This deserves closer attention: Step (4) of Algorithm 4.1 allows, in principle, to unite fine-level agglomerates that are solely connected to the coarse-level agglomerate under construction via an edge. While this situation hardly even occurs for 2D (unstructured) meshes, it appears more frequently in the 3D case. However, this potential shortcoming can easily be fixed by restricting the set of candidates in the process of removing faces. In the above formulation any candidate $f_{k'}$ to be selected in Step (4) has to be a neighbor of $f_k$, i.e., $f_k$ and $f_{k'}$ have to be adjacent faces, or equivalently, the condition

\[
(4.1) \quad (k, k') \in \mathcal{R}_{F, F}
\]

has to be satisfied. If instead of (4.1) one requires

\[
(4.2) \quad (k, k') \in \mathcal{R}_{F, F} \cap \mathcal{R}_{F, A, F},
\]

then strong connectivity of fine-level agglomerates (connectivity via fine-level faces) is assured within each coarse-level agglomerate. We put this slight modification into effect.

Nevertheless, as already stated in [13], Algorithm 4.1, when applied to 3D meshes in a straightforward manner, does not produce the well-shaped agglomerates as it

\(^2\)We assume a non-degenerated topology, i.e., all relations being sparse in the usual sense.
A multilevel-topology concept does in the 2D case. In Figure 1 the resulting agglomeration history is illustrated by considering a 3D structured finite element mesh consisting of 5×5×5 hexagonal elements. For visualization we used the General Mesh Viewer (GMV), a software package developed by Frank Ortega [8].

In the next section, we will discuss some basic changes of the agglomeration procedure and propose an improved algorithm.

Figure 1. Agglomeration based on Algorithm 4.1: 5×5×5 hexagonal mesh

5. An improved agglomeration algorithm for 3D problems

The basic principle of Algorithm 4.1 is to put integer weights on the faces where the weight of each face reflects its priority for being removed in the next step. The integer-valued weight function \( w = w_F \) is then modified according to certain rules (Step (2) and Step (3)) and the construction of each agglomerate is carried out by
removing a (finite) sequence of adjacent faces with non-decreasing weights, thereby collecting all fine-level agglomerates being connected via these faces; it stops if the continuation of such a sequence of faces with non-decreasing weights is no longer possible in which case the construction of the next agglomerate starts. Looking at the sequences of faces arising in the 3D case one observes that they are in general too short in order to produce well-shaped agglomerates.

Our first attempt to adapt the algorithm was to modify its incrementation rules in adding a third incrementation step based on the face-vertex-face relation $R_{F,V,F}$. We examined various combinations of increments but none of them led to a significant improvement.

We then changed the agglomeration technique basically and finally obtained much better results, i.e., well-shaped agglomerates in three-dimensional space. The main point is, and that is what we suggest to be a proper generalization of Algorithm 4.1 for the 3D case, to construct agglomerates by removing edges rather than faces. Typically, the only remaining faces occur in regions close to the boundary (especially near corners); a simultaneous update of edge and face weights, however, provides the necessary information to complete the agglomeration process in all cases. That means, at a final stage, agglomerates are (possibly) constructed by removing the remaining faces.

The algorithm we propose now works with the topological relations:

- $R_{E_m,E_{m-1}}$ (cf., Equation (3.7))
- $R_{F_m,F_{m-1}}$
- $R_{E_m,F_{m-1}}$
- $R_{E_m,F_{m-1},E_{m-1}}$ (cf., Equation (3.8))
- $R_{F_m,F_{m-1}}$ (cf., Equation (3.3))
- $R_{A_m,F_{m-1}}$
- $R_{F_m,F_{m-1}}$
- $R_{F_m,F_{m-1}}$ (cf., Equation (3.4))

and utilizes the two weight functions $w_E : E \to \mathbb{Z}$ and $w_F : F \to \mathbb{Z}$, which assign integer weights to all edges $e_k \in E := E_{m-1} \setminus \partial E_{m-1}$ and faces $f_k \in F := F_{m-1} \setminus \partial F_{m-1}$, respectively.

**Algorithm 5.1.** [Agglomeration in 3D space].

- Initialize: Set $w_E(e_k) = 0$ for all edges $e_k \in E := E_{m-1} \setminus \partial E_{m-1}$; set $w_F(f_k) = 0$ for all faces $f_k \in F := F_{m-1} \setminus \partial F_{m-1}$; set $n_m = 0$; set Case='Edge';
- Global search: Find an edge $e_k \in E$ with maximal weight $w_E(e_k) \geq 0$;
  If there is no interior edge left (i.e., $w_E(e_k) < 0 \ \forall k$) then set Case='Face' and proceed with finding a face $f_k \in F$ with maximal weight $w_F(f_k) \geq 0$;
  increment $n_m := n_m + 1$; set $a_{nm} = 0$;
- Collect the level-$(m-1)$ agglomerates $a_{r_{m-1}} \in A := A_{m-1}$ that form $a_{nm}$;
  1. If Case='Edge': set $a_{nm} := a_{nm} \cup \bigcup_{p_m-1:(k,p_m-1) \in R_{E,A}} a_{p_m-1}$, and set $w_{E,\max} = w_E(e_k)$, $w_E(e_k) = -1$;
If Case='Face': set $a_{n_m} := a_{n_m} \cup a_{p_m-1} \cup a_{q_m-1}$ where $(k,p_m-1)$ and $(k,q_m-1)$ in $R_{F,A}$, and set $w_{F,max} = w_{F}(f_k)$, $w_{F}(f_k) = -1$;
(2) If Case='Edge': increment $w_{E}(e_i):= w_{E}(e_i) + 1$ for all edges $e_i$ with $i$ such that $w_{E}(e_i) \neq -1$ and $(k,i) \in R_{E,E}$;
If Case='Face': increment $w_{F}(f_i):= w_{F}(f_i) + 1$ for all faces $f_i$ with $i$ such that $w_{F}(f_i) \neq -1$ and $(k,i) \in R_{F,F}$;
(3) If Case='Edge': increment $w_{E}(e_j):= w_{E}(e_j) + 1$ for all edges $e_j$ with $j$ such that $w_{E}(e_j) \neq -1$ and $(k,j) \in R_{E,E} \cap R_{E,F}$;
If Case='Face': increment $w_{F}(f_j):= w_{F}(f_j) + 1$ for all faces $f_j$ with $j$ such that $w_{F}(f_j) \neq -1$ and $(k,j) \in R_{F,F} \cap R_{F,A}$;
(4) If Case='Edge': increment $w_{F}(f_j):= w_{F}(f_j) + 1$ for all faces $f_j$ that are adjacent to any face $f_i$ to which $e_k$ is an edge of, i.e., $(k,i) \in R_{E,F}$ and $(i,j) \in R_{F,F}$.
(5) If Case='Edge': choose an edge $e_{k'}$ with a maximal weight $w_{E}(e_{k'})$ subject to $(k,k') \in R_{E,E} \cap R_{E,F}$;
- If $w_{E}(e_{k'}) \geq w_{E,max}$: set $k' := k'$ and go to step (1);
- If $w_{E}(e_{k'}) < w_{E,max}$: the level-$m$ agglomerate $a_{n_m} \in A_m$ is complete;
set $w_{E}(e_k) = -1$ for all edges $e_k$ of level-$(m-1)$ agglomerates $a_{r_m-1}$ contained in $a_{n_m}$;
set $w_{F}(f_k) = -1$ for all faces $f_k$ of level-$(m-1)$ agglomerates $a_{r_m-1}$ contained in $a_{n_m}$;
go to step Global search;
If Case='Face': choose a face $f_{k'}$ with a maximal weight $w_{F}(f_{k'})$ subject to $(k,k') \in R_{F,F} \cap R_{F,A}$;
- If $w_{F}(f_{k'}) \geq w_{F,max}$: set $k' := k'$ and go to step (1);
- If $w_{F}(f_{k'}) < w_{F,max}$: the level-$m$ agglomerate $a_{n_m} \in A_m$ is complete;
set $w_{F}(f_k) = -1$ for all faces $f_k$ of level-$(m-1)$ agglomerates $a_{r_m-1}$ contained in $a_{n_m}$;
go to step Global search;

The distinction of an ‘Edge’ case and a ‘Face’ case in the above algorithm corresponds to its two phases: in the first phase coarse-level agglomerates are constructed by collecting fine-level agglomerates around edges that are removed based on their weights; in the second phase, when there are no interior edges left, the procedure corresponds with Algorithm 4.1. However, the two phases are coupled via Step (4) which increments certain face weights during the first phase (the ‘Edge’ case) in order to produce a proper initial distribution of face weights for the second phase (the ‘Face’ case).

Agglomeration based on removing edges essentially proceeds faster, that means, on average a sequence of adjacent edges links more fine-level agglomerates than a sequence of adjacent faces if both are of the same length. The first phase of Algorithm 5.1 can also be viewed as a process of agglomerating fine-level faces (a whole bunch for each removed edge) and simultaneously collecting all fine-level agglomerates that are connected via any of these faces.
Figure 2 illustrates the agglomeration history for the same example as considered in Figure 1 but now using Algorithm 5.1. Note that the oblong agglomerates in the corners (2×1×1 blocks in the upper right picture and 4×1×1 blocks in the lower left picture), which are produced in the second phase of the algorithm, can suitably be matched with the other agglomerates (produced in the first phase) only if the corresponding ‘docking’ faces are incremented as it happens in Step (4). In general, Algorithm 5.1 maintains the structure of hexagonal meshes, as illustrated in Figure 3.

6. APPLICATION TO UNSTRUCTURED TETRAHEDRAL FINITE ELEMENT MESHES

Since we are not primarily interested in partitioning structured meshes we will put our emphasis on unstructured meshes in this section.

Before that, let us comment on two minor modifications of Algorithm 5.1 that balance the ‘size’ of the arising agglomerates: As in the 2D case (considering Algorithm 4.1), typically, a certain portion of fine-level agglomerates (we observed up to ten percent in the 3D case) is not fused by the agglomeration process, i.e., some of the resulting coarse-level agglomerates contain only one fine-level agglomerate. Further, some coarse-level agglomerates may become relatively large. (For tetrahedral meshes we observed up to about 30 level-0 agglomerates contained in the largest level-1 agglomerates.)
In order to find a remedy, we suggest:

- Abort the construction of agglomerates if their size exceeds a maximal size \( \text{MaxSize} \sim 10 \) (or 15).
- Append agglomerates of ‘size one’ in a post-processing step.

In case of abortion we continue with constructing the next agglomerate omitting the step ‘Global search’ since an edge (or a face) with maximal weight is already available.
Agglomerates of ‘size one’ we used to append according to a ‘best fit’ strategy, i.e., append them to a neighbor with which they share a maximal number of faces, thereby preferring the smaller neighbors. These modifications can easily be implemented and do not severely increase the overall costs. The sequence of nested agglomerates produced by the (modified) Algorithm 5.1 is shown in Figure 4 for a tiny 3D tetrahedral mesh with 240 elements.

We finally want to present the results for three unstructured FE meshes of more realistic size: the first, referred to as ‘small mesh’ comprising 2622 tetraeders, the second ‘medium mesh’ 24979 and the third ‘large mesh’ 252365 tetrahedral elements (level-0 agglomerates). Tables 1 to 3 list the number of agglomerates, faces, and edges as the agglomeration proceeds over the levels. In the rightmost column the minimal number of faces per agglomerate is included. We watched this quantity because if it dropped to one this would indicate an enclosure of one agglomerate by another. However, this situation was never observed when agglomeration was performed with Algorithm 5.1, even without limiting the maximal size of the generated agglomerates. As one can see from the tables, the reduction factor for the number of agglomerates from level zero to level one is close to 8 (the optimal value we observed for structured meshes) and it decreases to a value of 3 when we proceed to coarser levels, i.e., the agglomeration is reasonably fast. A visualization of the agglomeration history for the ‘medium mesh’, illustrating the formation of well-shaped agglomerates, is shown in Figure 5.

**Table 1. Agglomeration history for unstructured ‘small mesh’**

<table>
<thead>
<tr>
<th>level(k)</th>
<th>(#a)</th>
<th>(#f)</th>
<th>(#e)</th>
<th>(\text{min}(#f/#a))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2622</td>
<td>5519</td>
<td>3484</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>347</td>
<td>1511</td>
<td>1677</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>74</td>
<td>428</td>
<td>633</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>122</td>
<td>211</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>22</td>
<td>39</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>6</td>
<td>12</td>
<td>6</td>
</tr>
</tbody>
</table>

**Table 2. Agglomeration history for unstructured ‘medium mesh’**

<table>
<thead>
<tr>
<th>level(k)</th>
<th>(#a)</th>
<th>(#f)</th>
<th>(#e)</th>
<th>(\text{min}(#f/#a))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>24979</td>
<td>51325</td>
<td>31230</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3153</td>
<td>14000</td>
<td>15196</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>635</td>
<td>3796</td>
<td>5461</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>160</td>
<td>1056</td>
<td>1807</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>43</td>
<td>289</td>
<td>552</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>67</td>
<td>131</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>17</td>
<td>31</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>6</td>
<td>12</td>
<td>6</td>
</tr>
</tbody>
</table>
Figure 4. Agglomeration based on Algorithm 5.1: unstructured mesh with 240 elements

Table 3. Agglomeration history for unstructured ‘large mesh’

<table>
<thead>
<tr>
<th>level $k$</th>
<th>$#a$</th>
<th>$#f$</th>
<th>$#e$</th>
<th>$\min(#f/a)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>252365</td>
<td>511085</td>
<td>304096</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>31768</td>
<td>141221</td>
<td>150666</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>7513</td>
<td>46362</td>
<td>64519</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1836</td>
<td>13386</td>
<td>22467</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>465</td>
<td>3639</td>
<td>6950</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>113</td>
<td>866</td>
<td>1837</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>31</td>
<td>230</td>
<td>548</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>59</td>
<td>146</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>18</td>
<td>39</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>6</td>
<td>12</td>
<td>6</td>
</tr>
</tbody>
</table>
7. Conclusions

We refined the sparse matrix element topology presented in [11] by introducing edges (as topological elements), and described an agglomeration-based multilevel-topology concept for general three-dimensional finite element meshes. We presented efficient procedures for the setup of certain topological relations and developed a generalization of the agglomeration algorithm proposed in [3]. The resulting method, which, in practice, is of linear complexity (in the number of edges), can be used...
in various areas of finite element computation, e.g., as a mesh partitioner for load balancing problems or domain decomposition methods. Several experiments with structured and unstructured 3D finite element meshes, demonstrating the proper performance of the improved agglomeration algorithm, have been presented.

Based on the provided topological information we currently develop an Edge-Element AMG method for Maxwell's equations. Further investigations are planned on the utilization of the proposed concept in the area of element-agglomeration AMGe (Algebraic MultiGrid based on element interpolation) with application to 3D elasticity problems.

**Acknowledgements.** The authors would like to acknowledge the stimulating discussions with Panayot Vassilevski who also gave valuable hints for interesting future applications of the presented concept.

**References**


