Engineering with Computers (2008) 24:59–78
DOI 10.1007/s00366-007-0069-7

ORIGINAL ARTICLE

A general topology-based framework for adaptive insertion of cohesive elements in finite element meshes

Glaucio H. Paulino · Waldemar Celes ·
Rodrigo Espinha · Zhengyu (Jenny) Zhang

Received: 4 August 2006 / Accepted: 13 September 2006 / Published online: 24 July 2007

Abstract Large-scale simulation of separation phenomena in solids such as fracture, branching, and fragmentation requires a scalable data structure representation of the evolving model. Modeling of such phenomena can be successfully accomplished by means of cohesive models of fracture, which are versatile and effective tools for computational analysis. A common approach to insert cohesive elements in finite element meshes consists of adding discrete special interfaces (cohesive elements) between bulk elements. The insertion of cohesive elements along bulk element interfaces for fragmentation simulation imposes changes in the topology of the mesh. This paper presents a unified topology-based framework for supporting adaptive fragmentation simulations, being able to handle two- and three-dimensional models, with finite elements of any order. We represent the finite element model using a compact and "complete" topological data structure, which is capable of retrieving all adjacency relationships needed for the simulation. Moreover, we introduce a new topology-based algorithm that systematically classifies fractured facets (i.e., facets along which fracture has occurred). The algorithm follows a set of procedures that consistently perform all the topological changes needed to update the model. The proposed topology-based framework is general and ensures that the model representation remains always valid during fragmentation, even when very complex crack patterns are involved. The framework correctness and efficiency are illustrated by arbitrary insertion of cohesive elements in various finite element meshes of self-similar geometries, including both two- and three-dimensional models. These computational tests clearly show linear scaling in time, which is a key feature of the present data-structure representation. The effectiveness of the proposed approach is also demonstrated by dynamic fracture analysis through finite element simulations of actual engineering problems.

Keywords Fragmentation simulation · Cohesive zone models (CZM) · Intrinsic model · Extrinsic model · Topological data structure

G. H. Paulino (✉) · Z. (Jenny) Zhang
Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, Newmark Laboratory, MC-250, 205 North Mathews Avenue, Urbana, IL 61801-2397, USA
e-mail: paulino@uiuc.edu

W. Celes · R. Espinha
Tecgraf/PUC-Rio—Computer Science Department, Pontifical Catholic University of Rio de Janeiro, Rua Marquês de São Vicente 225, Rio de Janeiro, RJ 22450-900, Brazil

1 Introduction

Fracture and fragmentation, including dynamic events [1, 2] have a broad range of engineering applications [3]. In general, such phenomena involve switching from a continuum to a discrete discontinuity, which can be investigated by means of a cohesive zone model (CZM) of fracture. The CZM can be computationally simulated either by enrichment functions [4] or inter-element techniques [5, 6]. The latter is the approach of choice in this work, i.e., cohesive elements are inserted between bulk (or volumetric) finite elements. We distinguish two types of cohesive elements: intrinsic and extrinsic. In the former case, the cohesive elements are inserted a priori, i.e., at the pre-processing stage (before the simulation starts). In the latter case, the cohesive elements are inserted adaptively during the course of the finite element analysis, i.e., the cohesive elements are inserted when needed, and where needed. The present data structure handles both types of cohesive elements, either in two-dimensional (2D) or three-dimensional (3D) models, with Lagrangian-type finite elements of any order.

During the course of extrinsic fragmentation analysis, new cohesive elements are adaptively inserted at element interfaces, thus imposing changes in the topology of the mesh. In order to efficiently treat these topology changes, access to a topological data structure is needed. In fact, several finite element applications, especially adaptive analysis, require a topological data structure that provides adjacency information among the mesh topological entities [7–14]. For instance, in adaptive fragmentation simulations, element connectivity varies as new cohesive interfaces are inserted. The insertion of a cohesive element may require the duplication of nodes. Whether a node has to be duplicated depends on the topological classification of the facet along which the cohesive element has to be inserted [15–17]. Intrinsic CZMs can also benefit from a topology-based framework. Although the regions where to insert the cohesive elements are chosen in advance, the support of a topological data structure allows the use of conventional mesh generators to obtain the initial bulk element mesh. The cohesive elements can then be automatically inserted within the selected regions (e.g., along a line or a rectangular area for 2D applications; and along a plane or a volumetric region for 3D applications).

The storage space usually required by a complete topological data structure can become a crucial problem [7, 8] for finite element mesh representation. To address this problem, we have introduced a new compact adjacency-based topological data stucture, denoted by TopS, for finite element mesh representation—see our previous work [17, 18]. The proposed data structure uses reduced storage space while being complete, in the sense that it preserves the ability to extract all topological adjacency relationship among the mesh entities in time proportional to the number of retrieved entities.

In this paper, we explore the use of TopS, the specific topological data structure proposed in [17, 18], for supporting fracture and fragmentation simulations using cohesive zone modeling. We introduce a new topology-based algorithm that systematically classifies fractured facets (i.e., facet along which fracture has occurred), and identifies the topological modifications needed to preserve model consistency. The algorithm follows a set of procedures that consistently perform all the topological changes needed to update the model. The same set of procedures can be used for any type of finite element mesh, for both 2D and 3D models, including linear, quadratic, and higher order elements. One of the main advantages of this approach is that the same topological framework is employed for consistently supporting a variety of fragmentation simulations, despite the model dimension and the element order.

This paper is organized as follows. Section 2 discusses the interface models for cohesive fracture. Section 3 discusses previous works on topological data structures for mesh representation and for fragmentation simulations. Section 4 describes TopS, the topological data structure framework adopted in this work. Section 5 illustrates elements that are presently available in the data structure and shows how to incorporate a new element using the concept of element template. Section 6 (conceptually, the main section of the paper) addresses topological classification of facets, which define the fracture path. Section 7 provides many numerical results demonstrating that the elapsed time scales linearly with respect to the number of cohesive elements inserted. In Sect. 8, applications of the proposed framework in actual dynamic fracture analysis are demonstrated through two finite element simulations, one in 2D and the other in 3D. Finally, concluding remarks and suggestions for future work are given in Sect. 9.

2 Intrinsic and extrinsic cohesive models

Cohesive models are versatile and effective tools for numerical simulation of various separation phenomena in solids (e.g., fracture and fragmentation) [19–21]. There are two basic types of CZMs: intrinsic and extrinsic. The main distinction between them is the presence of the initial elastic curve, as shown in Fig. 1. Intrinsic CZMs assume that traction T first increases with increasing interfacial separation δ, reaches a maximum value, then decreases and finally vanishes at a characteristic separation value δ_c, where complete decohesion is assumed to occur. On the other hand, extrinsic CZMs assume that separation only occurs when the interfacial traction reaches the finite strength, and once the separation occurs, the interfacial cohesion force monotonically decreases as separation increases, and finally vanishes at the characteristic separation value δ_c.

In general, the so-called intrinsic models assume that all cohesive elements are embedded in the discretized structure prior to the beginning of the simulation, and thus the mesh connectivity remains unchanged during the whole simulation process [5, 22–26]. From the mesh representation point of view, intrinsic models allow easy implementation. However, they introduce artificial compliance, which depends on the area of cohesive element surfaces introduced and the cohesive law parameters relative to bulk element property, e.g., the ratio of cohesive strength to elastic modulus [19]. If the crack grows along a pre-defined path with relatively few number of cohesive elements, the adverse effect is relatively minor; while for simulations involving cohesive elements inserted in a large area or volume, the adverse effect can be more pronounced and lead to convergence problems in implicit simulations [22].

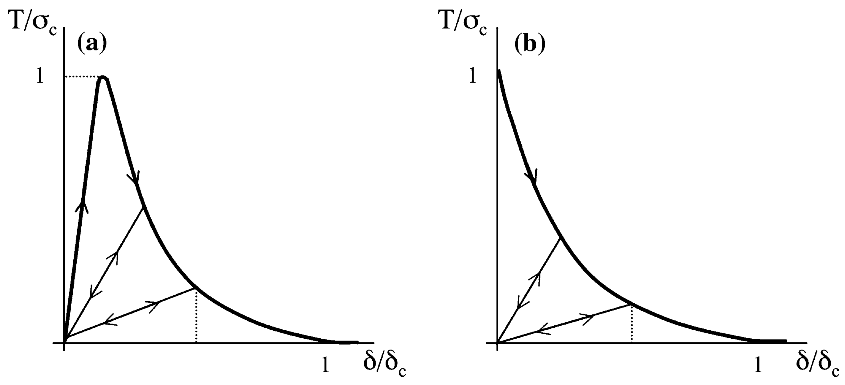


Fig. 1 Comparison of two typical cohesive zone models (*CZMs*): **a** intrinsic CZM; **b** extrinsic CZM

On the other hand, the so-called extrinsic models require adaptive insertion of cohesive elements (in space and time) in the finite element mesh [27]. This usually requires an elaborate updating scheme for the modified mesh by inserting new nodes and interface elements [16, 17]. Therefore, the geometrical and topological (e.g., connectivities) information need to be updated as the simulation progresses. The extrinsic model avoids the artificial softening effect present in intrinsic models.

The above observations regarding intrinsic and extrinsic models are of a general nature, and both models are of interest in this work. Different types of CZMs within each group are developed based on various considerations, and one may be preferred over the other depending on the problem. TopS, the topological data structure used in this work, applies to both models. The advantage for extrinsic models is obvious. In regard to intrinsic models, the data structure can be used to generate cohesive elements within pre-specified regions of an existing mesh. For instance, in assessing cracks in materials with microstructure, the data structure can be of great help in generating intrinsic elements around particles [28] or grain boundaries [29].

3 On topological data structures

Topological data structures define a model by means of a set of topological entities. These entities should provide an unambiguous abstraction of the underlying model [17, 30–32]. For finite element mesh representation, the usually defined topological entities include *region*, *face*, *edge*, and *vertex* [7–10]. Regions are 3D entities bounded by a set of faces, which are 2D. Edges are one-dimensional entities that delimit the faces, and vertices are zero-dimensional entities that represent the boundary of edges and are associated to the mesh nodes. For 3D meshes, each region corresponds to a finite element and internal faces represent the interface between two elements. For 2D meshes, each face corresponds to a finite element and internal edges represent the interface between elements.

Previous works have proposed the use of topological data structures, originally designed for solid modeling [30–32], to support fracture simulation [33, 34]. The result is a highly integrated framework where mesh entities are related to the corresponding geometric model entities, but such topological data structures impose a prohibitive cost of storage space for large finite element meshes. In order to minimize the required storage space, researchers have been engaged in the development of reduced (compact) topological data structures [7, 8]. The main idea consists of not explicitly representing all the topological entities. Representation of *implicit entities are created, as required, "on-the-fly"*. This approach tends to reduce the storage space needed to represent the mesh but faces the challenge of appropriate management of implicit entities in order to maintain the model consistency [7]. As an alternative solution, some topological data structures have been specifically designed for attending the needs of particular mesh generation algorithms [12, 13] and analysis applications [14]. In references [9, 10], a different approach has been proposed. Instead of using a particular data structure, the authors have presented an algorithm-oriented mesh database. They are then able to dynamically extract different mesh representations according to the application needs in terms of adjacencies. In general, this is achieved by extracting and explicitly storing the adjacency relationships needed by a particular application, resulting in efficient representations for querying operations. However, this approach still faces the challenge of keeping the whole data structure consistent when modifying the mesh. Moreover, explicitly storing all the needed sets of adjacencies may demand a large amount of memory space (see Garimella [8] for a few comparisons). It might be better to efficiently extract each relationship just when required.

Previous reduced topological data structures have represented implicit faces and edges by their bounding nodes [7–10], but those representations are ambiguous for fragmentation simulation. Finite element models for fragmentation simulation require the representation of cohesive elements [15, 16, 35, 36], which may have distinct edges with the same bounding nodes. Pandolfi and Ortiz [15, 16] have proposed a topological data structure for supporting fragmentation simulations. Their work focused specifically on representing quadratic tetrahedral meshes. They have opted for explicitly representing all topological entities and their corresponding adjacency relationships, thus ending up with data structures that require large amounts of storage space.

Notice that the use of region, face, edge, and vertex as the defined topological entities imposes challenges for generalizing previous proposals [7–10, 33, 34]. If this set of topological entities is used for designing a unique topological framework to represent both 2D and 3D meshes, one faces the problem of managing different semantics associated to faces and regions. Faces, in 2D, represent finite elements, while in 3D they represent the boundary of finite elements. Similarly, in 3D, regions represent elements but are meaningless for 2D meshes. It would be better if the semantics associated to each topological entity did not change, despite the mesh dimension. This is a major contribution of TopS [17, 18], which is employed here to support the topological framework for fragmentation simulations.

4 Adjacency-based topological data structure (TopS)

In order to support both intrinsic and extrinsic fragmentation simulations, we use a novel compact adjacency-based topological data structure (TopS) that requires small storage space while providing access to all adjacency relationships in time proportional to the number of retrieved entities. This basic data structure is presented in detail in references [17, 18]. Here, we briefly describe its main concepts and focus on its use for supporting large scale finite element fragmentation simulations.

4.1 Topological entities

TopS was designed to represent meshes with any type of elements defined by templates of ordered nodes. Therefore, it defines a new set of topological entities, which is used to describe both 2D and 3D models. It explicitly represents only two topological entities: *element* and *node*. The *element* is an abstract entity that represents finite elements (any type and order). The node represents finite element nodes (both corner and mid-side nodes). Each element has references to its boundary nodes and to the adjacent elements. The element is specialized for each type of finite element. For instance, a quadratic tetrahedral finite element would have ten boundary nodes and four adjacent elements. Each node, besides its position, keeps a reference to one of its incident elements.

TopS also defines and implicitly represents other topological entities. *Facet* represents the interface between two elements. *Edge* represents a one-dimensional entity and is bounded by two vertices. *Vertex* represents a corner node (there is no vertex associated to a mid-side node). Thus, essentially, vertex is a node, but not every node is a vertex. An important point to note is that, for being able to handle both 2D and 3D meshes, the facet always represents the interface between two elements. Thus, for 3D models, a facet corresponds to a 2D entity and is defined by a cyclic set of edges. For 2D meshes, it corresponds to a one-dimensional entity and is defined by a unique edge. Facets resting on the model boundary have only one interfacing element.

In order to facilitate retrieving topological adjacency relationships, TopS also defines and implicitly represents oriented topological entities, namely *facet-use*, *edge-use*, and *vertex-use*, associated to the use of facets, edges, and vertices by an element, respectively. Each finite element in isolation is composed by a set of facets, edges, and vertices. These local entities are labeled by identification numbers (id's). The topology of each element is known in advance and depends only on the element type (e.g., T3, T6, Tetra4, and Tetra10). Consequently, for each type of element, we define its *element template* [7]. Based on an element template, we can extract all adjacency relationships relating the local entities of such an element type. The local facets, edges, and vertices of an element in isolation correspond, respectively, to the use of facets, edges, and vertices of the mesh by that element. Therefore, the element template provides access to adjacency relationships relating entity-uses within an element.

Element and node are *explicitly* represented by abstract types. Each element or node has to be allocated and included in the model. Facet, edge, vertex, and their associated uses, are *implicitly* represented and retrieved, when required, *"on-the-fly"*. To easily handle these implicit entities, TopS represents them as *concrete types*, being treated as ordinary values, similarly to any other built-in type of a programming language. By using concrete types, we avoid the need of dynamic allocation while manipulating implicit entities: they are returned by value and allocated on the stack, which greatly simplifies managing entity uniqueness and lifetime. Each implicit entity is represented by integer values packed in a 32-bit word [18]. Each facet-use, edge-use, or vertex-use is represented by a reference to the element (Ei) using the entity and a local number (id) that identifies the corresponding entity within the element (Ei, id). Each facet, edge, or vertex is simply represented by one of its uses. One should note that by representing edges and facets in this way, two different edges (or facets) sharing the same bounding nodes, a quite common topological configuration in fractured models, have different representations because they are associated with different edge-uses (or facet-uses). A main drawback of previous reduced representations is to rely implicit-entity representations solely on the bounding nodes [7–10], thus being unable to distinguish the two different edges (or facets).

The diagram in Fig. 2 shows the entities represented by the data structure, together with the stored data and the directly retrieved topological adjacency. From the model, we have access to the set of nodes and elements that compose the mesh. From each element, we can directly access the adjacent elements and the boundary nodes. Conversely, from each node, we access an incident element. Based on the element template, from an element, we can also access its associated facet-uses, edge-uses, and vertex-uses, thus having access to facets, edges, and vertices. From each entity-use, we can also access its boundary nodes.

4.2 Adjacency relationships

There are a total of 25 adjacency relationships among the five defined topological entities (element, node, facet, edge, and vertex), as illustrated by Fig. 3. As shown by Celes et al. [17], the data structure is capable of retrieving any of these relationships in time proportional to the number of retrieved entities. Three of them are of particular interest for fragmentation simulations, and are marked by thicker lines in Fig. 3. These are the relationships that provide access to all the adjacent elements of a given facet, edge, or vertex.

Finding the adjacent elements of a given facet, edge, or vertex corresponds to retrieving all the uses associated to the given facet, edge, or vertex, respectively. Based on the data structure representation, together with the element templates, these sets of uses can be efficiently retrieved:

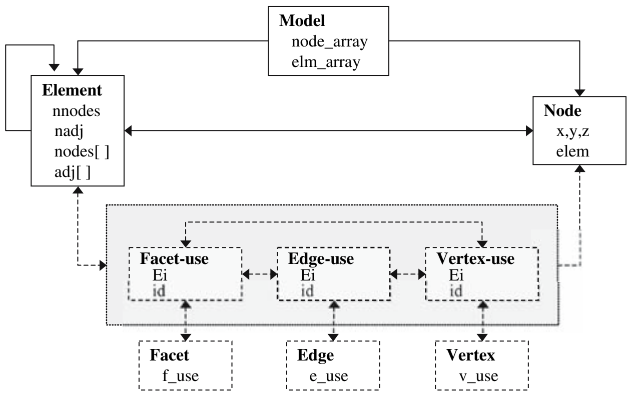


Fig. 2 Schematic representation of topological entities in the data structure (*TopS*). *Solid boxes* represent explicit entities and *dashed boxes* represent implicit entities. *Solid arrows* represent explicitly stored adjacency and *dashed arrows* represent implicitly stored adjacency, which are extracted based on the element templates

1. *Uses of a facet*: given a facet, we first access one of its uses from its representation. Then, with the referenced element and the local facet id, we access the other use of the same facet, by using the adjacency information stored in the element representation.
2. *Uses of an edge*: given an edge, we first access one of its uses from its representation. Then, based on the element template, we access the adjacent facet-uses within the element. By using the adjacency information stored in the element, we can reach each adjacent element and then access the uses of the corresponding edge. The procedure is repeated until all adjacent elements are visited.
3. *Uses of a vertex*: given a vertex, we first access one of its uses from its representation. Then, based on the element template, we access the set of adjacent facet-uses within the element. By using the adjacency information stored in the element, we can reach each adjacent element and then access the uses of the corresponding vertex. The procedure is repeated until all adjacent elements are visited.

It is important to note that these sets of uses are all retrieved based on the adjacency information provided by the element representation, which provides access to the adjacent elements. Once we have the set of uses, we can have access to the adjacent elements: it suffices to access the element associated with each one of the retrieved entity-uses (facet, edge, or vertex).

5 Finite elements for fragmentation simulation

In order to handle fragmentation simulation, we have added support for different types of finite elements in the data structure. Previous proposals [15, 16] have opted for identifying cohesive elements as attributes attached to the facets of bulk elements. We have opted for explicitly representing the cohesive elements. In this way, cohesive elements are treated as any other type of element, resulting in a more flexible and concise data structure. As an example of such flexibility, cohesive elements can hold application attributes, as any other element. Moreover, with respect to their representation in the data structure, the same bulk element type can be used for analysis other than fragmentation simulation. Similarly, the interface elements (between bulk elements) can be used for any analysis regarding interfacial behavior (e.g., non-cohesive).

Figure 4 illustrates a few types of finite elements added for fragmentation simulation. For instance, for 2D simulation, we have added support for both linear and quadratic triangular and quadrilateral element (T3, T6, Q4, and Q8), together with the two corresponding cohesive elements: the

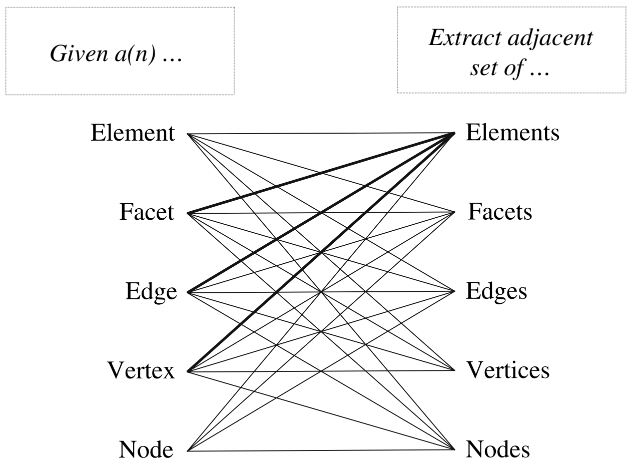


Fig. 3 The 25 topological adjacency relationships defined among the data structure topological data. Three of them are of special interest for fragmentation simulations: the two elements adjacent to a given facet, the set of elements adjacent to a given edge, and the set of elements adjacent to a given vertex

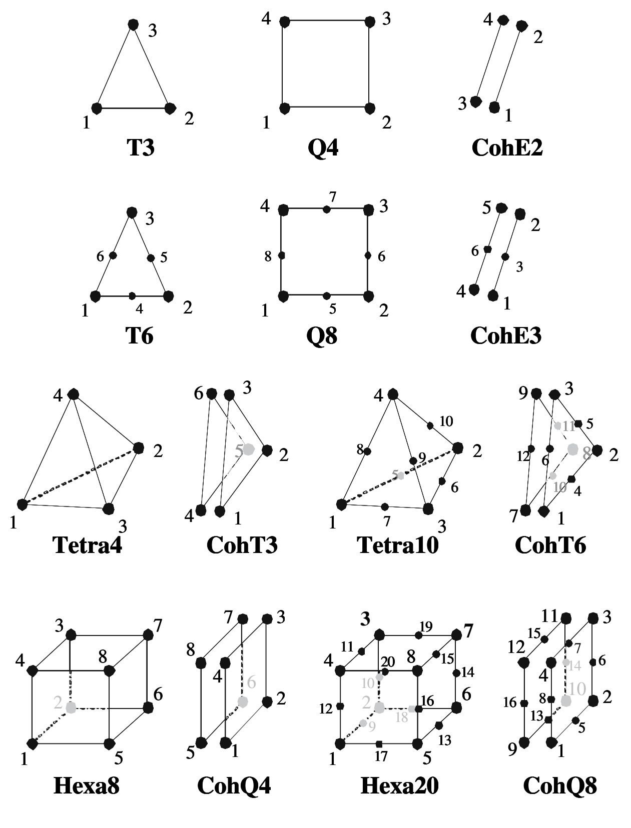


Fig. 4 Illustration of element types (both volumetric and cohesive) for fragmentation simulation

cohesive element with linear edge (CohE2) and the cohesive element with quadratic edge (CohE3). For 3D, we have added support for both linear and quadratic, tetrahedral and hexahedral elements (Tetra4, Tetra10, Hexa8, and Hexa20), and for their corresponding cohesive elements (CohT3, CohT6, CohQ4, and CohQ8, respectively). It is important to note that the cohesive elements internally represent the mesh boundary. Thus, their element templates dictate that there is only one local facet adjacent to anyone of their vertices or edges, even when opposite vertices share the same nodes. As an example, it is valid for a cohesive element of type CohE2 to have the following illustrative incidence: node$_A$, node$_B$, node$_C$, and node$_B$. This indicates that two vertices of the element share the same node. However, the element template dictates that the second vertex of the element is adjacent to a local facet while the fourth vertex is adjacent to a different local facet.

The concept of *element template* [7, 17] is quite flexible and versatile. For instance, if a new type of element is needed, it can be easily incorporated in TopS by means of its element template, which relates to the local topological entities of an element in isolation. This general concept holds for both bulk and interface elements.

6 Topological classification of facets

During the course of extrinsic fragmentation simulation, the analysis application identifies at which facets new cohesive elements are to be inserted, depending on the fracture criterion used by the simulation. The insertion of new cohesive elements imposes topological changes in the data structure. In order to identify which operations have to be done for updating the data structure, we need a criterion to topologically classify the fractured facets. In this section, we first review previous proposals restricted to quadratic tetrahedral elements and then introduce a new systematic topological classification that can be applied to any type of element.

6.1 Previous proposals

Pandolfi and Ortiz [15, 16] have proposed a set of rules to classify the fractured facet in order to perform the appropriate topological changes in the model. Their criterion is applied to quadratic tetrahedral elements and is based on the position of the facet with respect to the model boundary (external, internal, or created by cracks). More precisely, in their data structure, a facet is bounded by a set of segments (edges), and the criterion to classify a facet is based on the position of the segments of the facet with respect to the boundary. As illustrated in Fig. 5, four different cases are identified. In all cases the facet itself is duplicated. The remaining required topological operations vary according to the case [15, 16]:

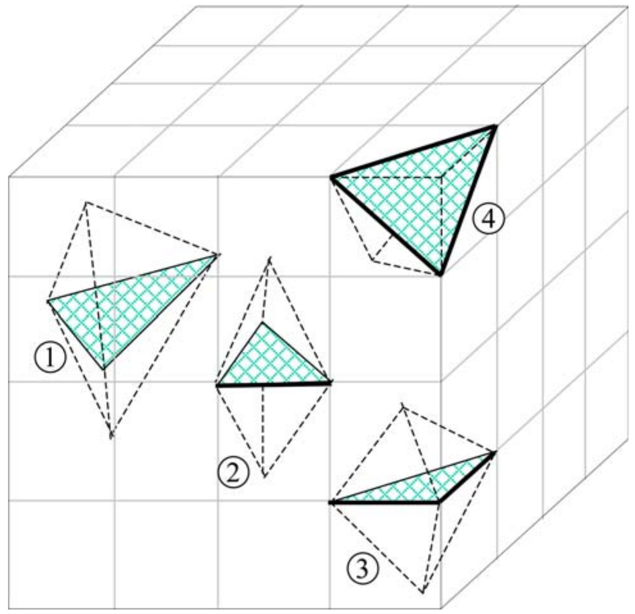


Fig. 5 Cases of facet classification identified by Pandolfi and Ortiz [15, 16] in an illustrative tetrahedral model (for simplicity, the diagonal edges on the mesh boundary are not represented). Fractured facets are dashed and segments on the boundary are in *bold*

- Zero segments on the model boundary: no further operation is required;
- One segment on the boundary: the segment (and its mid-side node) is duplicated;
- Two segments on the boundary: the segments (and their mid-side nodes) are duplicated; the corner-node shared by both segments is "a candidate to be" duplicated;
- All the three segments of the facet on the boundary: the segments (and their mid-side nodes) are duplicated and all three corner-nodes are "candidates to be" duplicated.

The approach by Pandolfi and Ortiz [15, 16] is restricted to quadratic tetrahedral elements. Although extension for tetrahedral elements of different order is straightforward, their criterion cannot be directly applied to other element types. For instance, for hexahedral elements, several other cases should be identified.

6.2 Systematic topological classification

In order to overcome the limitations of previous proposals, we introduce a *new* systematic topological classification that can be applied to any type of element, including both 2D and 3D elements. We define a set of procedures that, carried out step-by-step, consistently classify the facets, thus identifying the topological changes needed to update the data structure.

Once the analysis application identifies the facet where to include a new cohesive element, we have access to both associated facet-uses. Without loss of generality, let us name them as first facet-use (`fu1`) and second facet-use (`fu2`). Therefore, we can also name the interfacing elements: first element (`E1`) and second element (`E2`), as illustrated in Fig. 6a. The following procedures should then be carried out:

1. Insert the new cohesive element: the new cohesive element is created and inserted in the model, sharing facets with the two interfacing elements (`E1` and `E2`). Accordingly, the adjacency of both elements is updated to reference the new inserted element (Fig. 6b). Element E1 is no longer adjacent to element `E2`, and vice versa; they both now are adjacent to the new inserted cohesive element.
2. For each edge-use (`eu`) of the facet-use (`fu1`) associated to the first interfacing element (`E1`), do:
   - Starting at `eu`, retrieve all other uses of the same edge, based on the updated element adjacencies (Fig. 6c). If the edge-use associated to the `E2` is not reached (due to changes in the element adjacencies), duplicate the edge, which leads to duplicating all mid-side nodes, if they exist. If the `E2` is reached, the edge should not be duplicated.
3. For each vertex-use (`vu`) of the first facet-use (`fu1`) associated to the first interfacing element (`E1`), do:
   - Starting at `vu`, retrieve all other uses of the same vertex, based on the updated element adjacencies

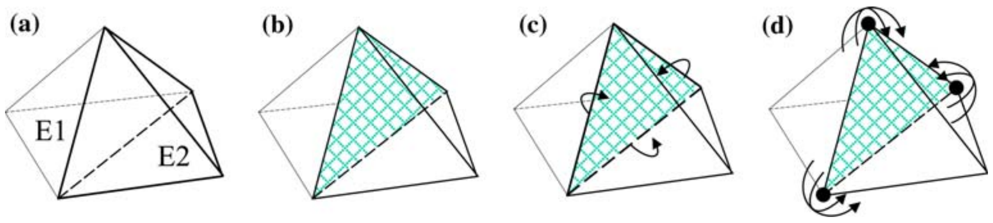


Fig. 6 Proposed procedures to classify the fractured facet: **a** two original adjacent tetrahedral elements; **b** the new cohesive element is inserted; **c** retrieval of elements around each edge starting at the first element, trying to reach the second one; **d** retrieval of elements around each vertex starting at the first element, trying to reach the second one

(Fig. 6d). If the vertex-use associated to the E2 is not reached (due to changes in the element adjacencies), duplicate the vertex, which leads to duplicating the associated corner node. If the E2 is reached, the vertex should not be duplicated.

Whenever a node is duplicated, element connectivity has to be updated. The new created node should replace the original node in all the "uses" reached by the retrieval process described above. In other words, all elements associated to the visited edge-uses (or vertex-uses) must have their incidence updated, and all element(s) not reached during the retrieval will continue referencing the original node.

The following pseudo-code illustrates the algorithm just described for inserting cohesive elements along fractured facet of bulk elements.

Pseudo-code: Procedures to insert a new cohesive element along a facet.

```
function InsertCohesive ( facet )

   fu1 = GetFacetUse ( facet, 1 )
   fu2 = GetFacetUse ( facet, 2 )
   E2 = GetElement ( fu2 )

   CreateCohesiveElement ( facet )

   for each edgeuse eu of fu1 do
      set = GetEdgesAdjacentToElement ( eu )
      if not IsInSet ( set, E2 ) then
         DuplicateEdge ( GetEdge ( eu ) )
      end if
   end for

   for each vertexuse vu of fu1 do
      set = GetVerticesAdjacentToElement ( vu )
      if not IsInSet ( set, E2 ) then
         DuplicateVertex ( GetVertex ( vu ) )
      end if
   end for

end function
```

6.3 Application to 3D meshes

The proposed set of procedures suffices for classifying all the cases identified by Pandolfi and Ortiz for tetrahedral quadratic meshes [15, 16]. In fact, it is easy to see that the procedures do correctly make all the required topological changes. Thus, Pandolfi and Ortiz work [15, 16] becomes a particular case of the present general criterion. Now let us consider the general case in 3D.

First, we should note that for an internal edge, after breaking the interface between the two interfacing elements, it is always possible to reach the E2 from the first one, rotating around the edge, that is, following the cyclic chain of edge-uses. Conversely, if the edge is resting on the model boundary, it is not possible to access the E2 from the first. Therefore, an edge on the boundary of a 3D model will always be duplicated. Figures 7 and 8 illustrate both situations together with the corresponding cross-section views.

Second, we have to investigate what happens to the surroundings of a vertex that is adjacent to a fractured facet. If the vertex is on the boundary of the model and shared by two adjacent edges of the facet that are also on the boundary, there is no way to access the E2 from the first one, after the adjacency between the two elements is broken. In this case, the vertex is duplicated. This situation is illustrated in Fig. 9, together with the effect of the topological changes. On the other hand, if the vertex is in the interior of the model or on the boundary, but shared by none or only one edge of the facet, it remains possible to reach the E2. These situations are illustrated in Fig. 10.

All the topological cases identified by Pandolfi and Ortiz [15, 16] fit well under this model. One should still note that the proposed procedures also apply for hexahedral elements, without any change, despite which edges of the facet rest on the boundary model. In fact, the extension to other elements is one of the advantages of the proposed topological classification.

6.4 Application to 2D meshes

The same set of procedures described above also work for 2D models. It is important to note that, in 2D, each facet is

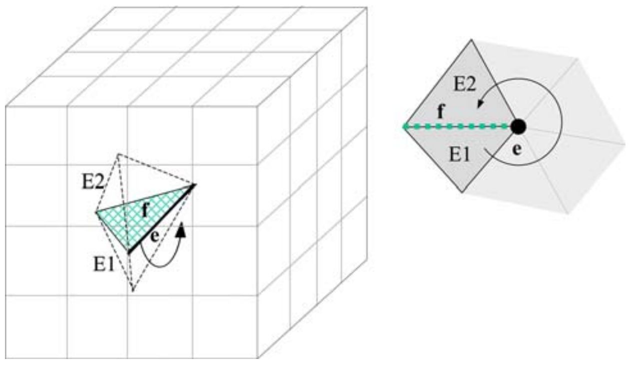


Fig. 7 Illustrative tetrahedral mesh (for simplicity, the diagonal edges on the mesh boundary are not represented). After the facet (*f*) is fractured, it remains possible to rotate around an internal edge (*e*) to reach the second element (*E2*) from the first one (*E1*)

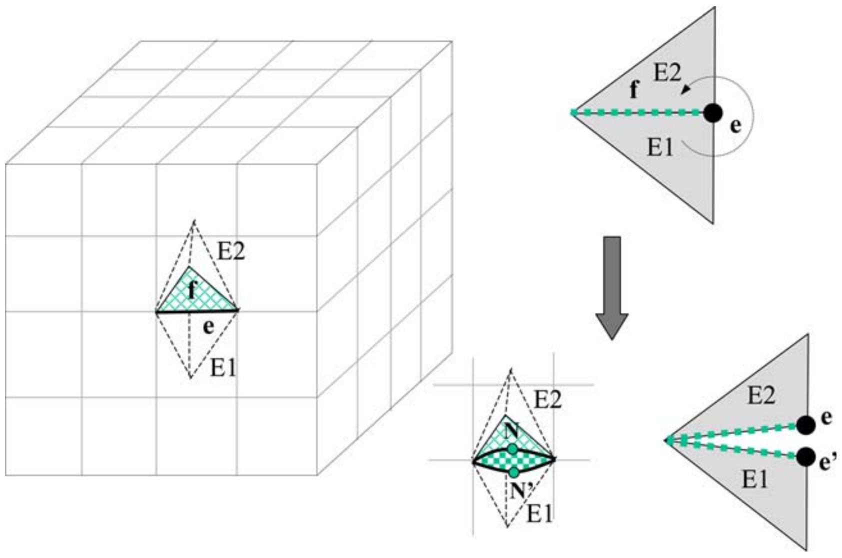


Fig. 8 Illustrative tetrahedral mesh (for simplicity, the diagonal edges on the mesh boundary are not represented). After the facet (*f*) is fractured, it is not possible to rotate around a boundary edge (*e*) to reach the second element (*E2*) from the first one (*E1*). As a result, the edge is duplicated (*e*′) and the eventual mid-side nodes (*N*) are duplicated (*N*′). In this case, the connectivity of elements is updated

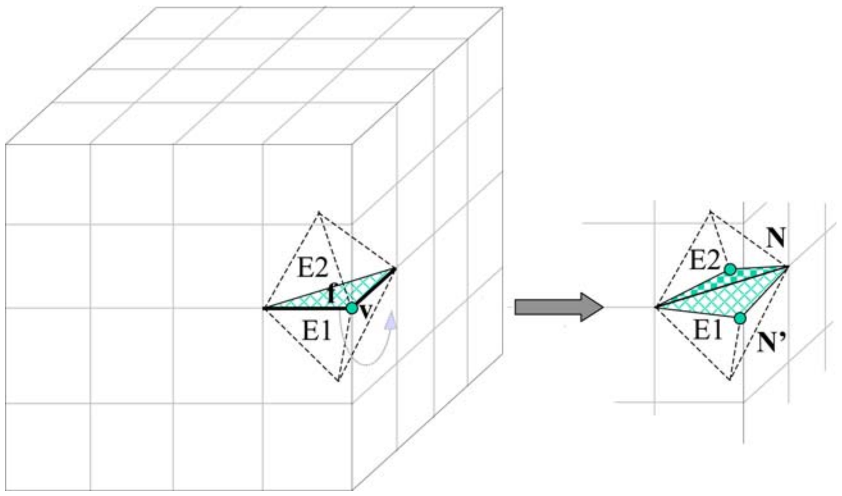


Fig. 9 Illustrative tetrahedral mesh (for simplicity, the diagonal edges on the mesh boundary are not represented). After the facet (*f*) is fractured, it is not possible to rotate around a boundary vertex (*v*) to reach the second element (*E2*) from the first one (*E1*) if the vertex is shared by two edges of the facet also on the boundary. As a result, the associated node (*N*) is duplicated (*N*′) and the connectivity of elements is updated

defined by a unique edge. Conversely, each edge-use of an element has only one facet-use adjacent to it. Therefore, once the interface between two elements is broken, there is no way to access the E2 starting at the associated edge-use of the E1. The unique facet-use is no longer adjacent to the E2. As a result, in 2D models, the edge associated to the fractured facet is always duplicated, and existing mid-side nodes, if any, are duplicated. Figure 11 illustrates three different configurations together with the resulting topological changes.

The procedure related to each vertex of the fractured facet may present different results. If it is an interior vertex, it is always possible to access E2 rotating around the vertex. However, if the vertex rests on the model boundary, E2 can no longer be reached, and the vertex has to be split (Fig. 11).

6.5 Avoiding non-manifold configurations

Pandolfi and Ortiz [15] have mentioned that, “inevitably, non-manifold situations, such as *shell* pinched at a point, do indeed arise during fragmentation.” We shall demonstrate that, under the topological framework proposed here (TopS), the model representation remains valid during fragmentation, even for complex crack patterns.

The topological data structure was designed to provide support for representing meshes with manifold domains. This means that the external boundary of a 3D mesh must have two-manifold topology; therefore, each edge on the boundary is shared by exactly two boundary faces. Accordingly, for 2D models, the external boundary must have one-manifold topology, with each vertex on the boundary having exactly two boundary edges connected to

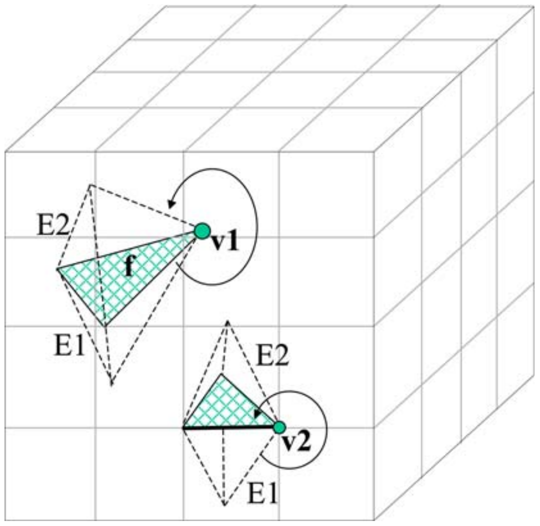


Fig. 10 Illustrative tetrahedral mesh (for simplicity, the diagonal edges on the mesh boundary are not represented). After the facet (*f*) is fractured, it remains possible to rotate around a vertex (*v1*) to reach the second element (*E2*) from the first one (*E1*), even if the vertex is on the boundary (*v2*), but not shared by two edges of the facet also on the boundary

it. Under these conditions, the data structure is complete, in the sense that it can efficiently retrieve all the adjacency relationships among the defined topological entities [17].

For a model discretized by finite elements, there are two types of non-manifold configurations, illustrated in Fig. 12, that are not presently supported by TopS. In the first configuration, two non-adjacent elements in 3D share the same edge. In the second, two non-adjacent elements, either in 3D or in 2D, share the same vertex. The former configuration represents a singularity at a non-manifold edge and the latter a singularity at a non-manifold vertex [37]. If neither of these two configurations occurs, the mesh representation is valid and complete. We should note that a set of disjoint elements represents a mesh with a manifold boundary. In fact, a fragmentation simulation may result in a set of disjoint manifold sub-meshes, each one composed by one or more connected elements.

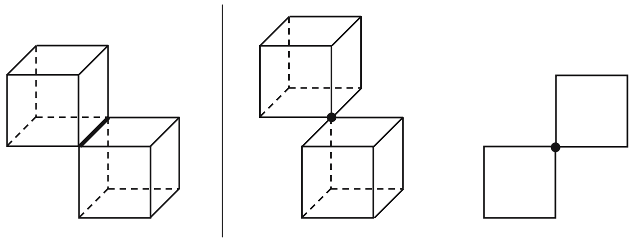


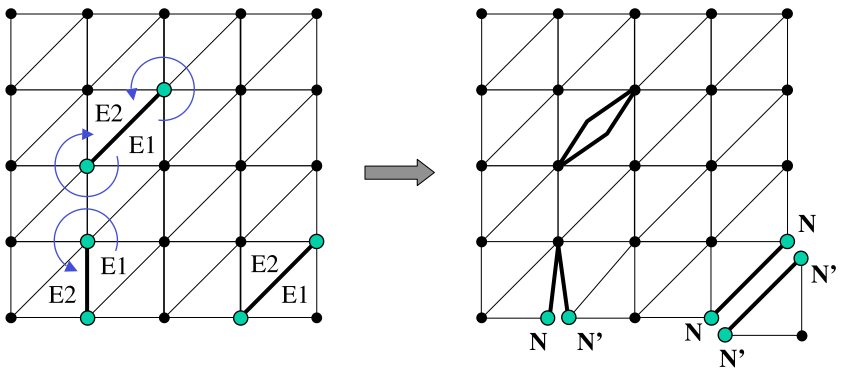
Fig. 12 Non-manifold configurations in 3D and 2D models. The *left* configuration represents a singularity at a non-manifold edge; the *other two* represent a singularity at a non-manifold vertex

During the course of the simulation, two non-adjacent elements may have nodes with the same geometric positions, thus having the same appearance as the configurations illustrated in Fig. 12. However, as long they are different nodes, the non-manifold configuration is not characterized. From a topological point of view, the two elements are disjoint.

As long as we start with a valid mesh representation, the proposed topological procedures to insert a cohesive element ensure that the adaptive model representation remains valid during the course of the simulation, even for complex crack patterns. In other words, the singularities illustrated in Fig. 12 do not arise while inserting cohesive elements in a mesh. A “singularity” at a non-manifold edge can only occur if we have more than one connected sub-mesh sharing the same edge. The second procedure (see Sect. 6.2) to classify fracture facets avoids such a configuration. After breaking the interface between the two elements, we check whether the elements around the edge remains connected. Whenever the connectivity is broken, we duplicate the edge, attaching the new edge to one of the connected set of elements. The third proposed procedure works in a similar way to avoid the occurrence of singularity at non-manifold vertices. The vertex is duplicated whenever the elements around it do not remain connected.

There is another non-manifold configuration that, in fact, may arise during the course of a fragmentation

Fig. 11 Fractured facets in 2D with their corresponding topological changes in an illustrative triangular mesh. Eventual mid-side nodes are not illustrated, but they would be duplicated whenever their associated edges are duplicated



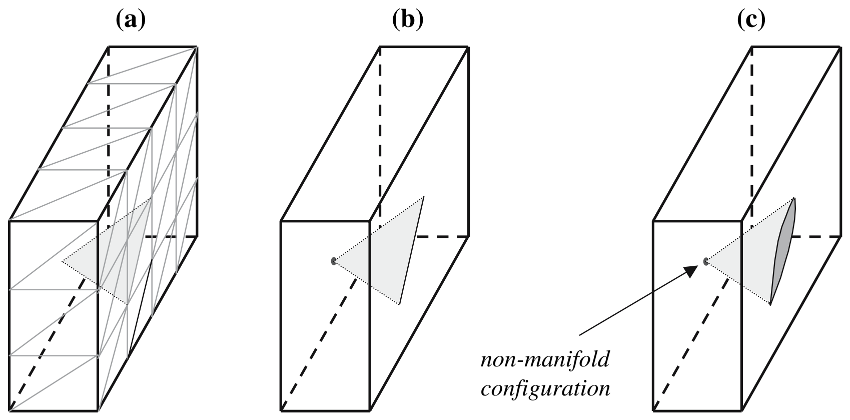


Fig. 13 A non-manifold configuration that may arise during the course of a fragmentation simulation. A new cohesive element is inserted along an internal facet of the model: **a** illustrative tetrahedral model with one of its internal faces highlighted in *gray*; **b** same model displaying the internal facet in isolation: one edge of the facet lies on one boundary of the model and the opposite vertex lies on another side of the mesh boundary; **c** model configuration after the insertion of a CZ element along the internal facet: the edge on the boundary is duplicated but the opposite vertex is not, thus a non-manifold configuration is characterized at this location. The mesh representation using TopS remains valid under this configuration

simulation. It occurs when a cohesive element is inserted along an internal facet whose vertices lie on different sides of the mesh boundary. Such an occurrence is illustrated in Fig. 13, where one edge of an internal triangular facet lies on one boundary of the model and thus is duplicated; the opposite vertex lies on another side of the mesh boundary. As this vertex is not duplicated, a non-manifold configuration is characterized at this location. Such a non-manifold configuration, however, does not invalidate the mesh representation because there is only one connected set of elements around the non-manifold vertex. Thus, the data structure remains complete under this configuration. In fact, according to the proposed topological classification, the node is not duplicated because it is possible to reach, from one element adjacent to the fracture facet, the second adjacent element. This criterion avoids the emergence of more than one connected set of elements around a vertex, which is the necessary condition for the data structure to be complete.

7 Computational experiments

We have run a set of computational experiments to test the scalability, efficiency, and correctness of the proposed algorithm to insert cohesive elements along the facet of bulk elements. In the experiments, we have considered a variety of models, from 2D to 3D, including both linear and quadratic meshes, thus demonstrating that the proposed approach is general and can be applied to a variety of models. The algorithm to insert cohesive elements is exactly the same, despite the model under consideration; that is, the code to implement the insertion of cohesive elements is the same, for 2D and 3D models, and for linear and quadratic meshes. This is the main advantage of using TopS: we achieve a unified topological framework for representing finite element models used on fragmentation simulations.

The basic model under consideration is a cylindrical specimen as illustrated in Fig. 14. Different 2D and 3D finite element meshes were generated to represent the cylindrical model, at different discretizations, using both linear and quadratic elements: T3, T6, Q4, Q8, Tetra4, Tetra10, Hexa8, and Hexa20. Figures 15–18 show the resulting meshes at illustrative discretizations.

For each generated mesh, we have tested the proposed framework decoupled from any mechanics simulation (setting up experiments similar to the one described by Pandolfi and Ortiz [15]). Cohesive elements were inserted, in a random order, at all the facets of the underlying meshes. The random order in which the cohesive elements are

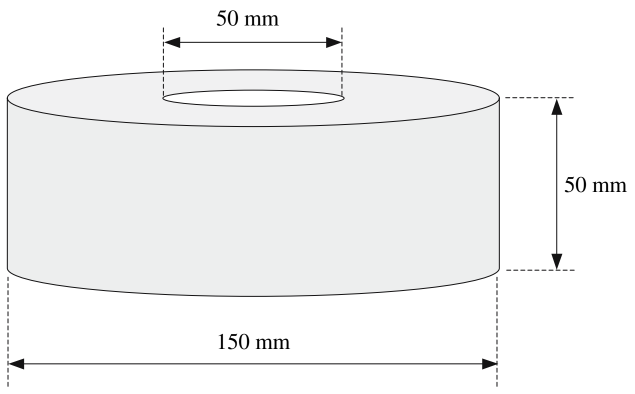


Fig. 14 Cylindrical model used in the computational experiments

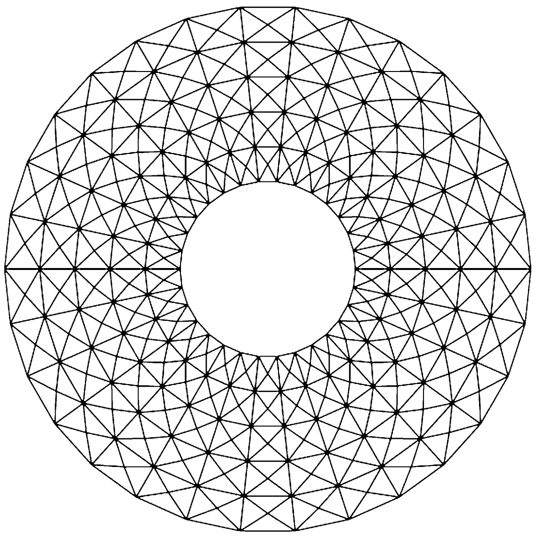


Fig. 15 Illustrative triangular mesh (for both T3 and T6) displaying a discretization of 5 × 30

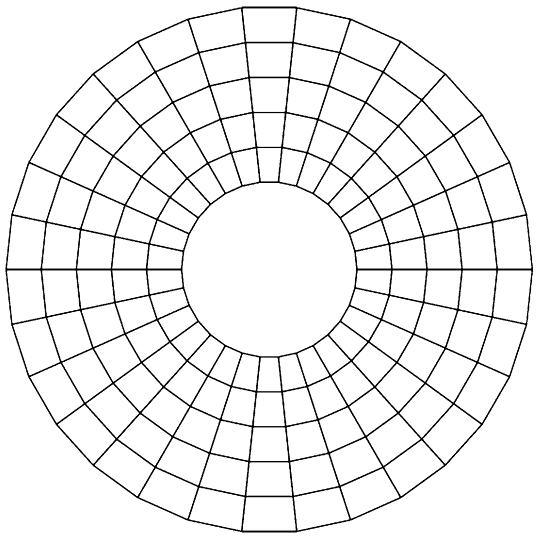


Fig. 16 Illustrative quadrilateral mesh (for both Q4 and Q8) displaying a discretization of 5 × 30

inserted results in arbitrarily complex crack patterns during the experiment. In the end, each node of the mesh is used by only one bulk element. We then have checked if the final obtained number of topological entities (nodes, elements, facets, edges, and vertices) were the expected ones. Figure 19 shows the achieved configuration of an illustrative hexahedral mesh after inserting cohesive elements, in a random order, along 20% of the facets of the model, illustrating the resulted arbitrary crack patterns. In Fig. 19b, we impose a separation in between each bulk element interface in order to better illustrate the mesh with the inserted cohesive elements (represented in blue).

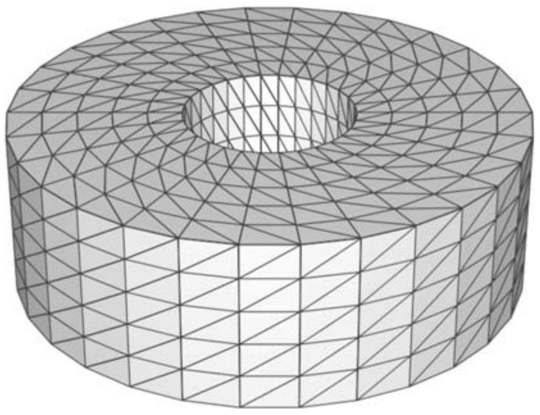


Fig. 17 General tetrahedral mesh (for both Tetra4 and Tetra10) displaying a discretization of 5 × 30 × 5

Table 1 presents the time needed to perform the insertion of cohesive elements at all bulk element interfaces.[1] For each type of mesh, we run the test for different mesh discretizations. Figures 20–23 plot the elapsed time against the number of cohesive elements inserted for the triangular, quadrilateral, tetrahedral, and hexahedral meshes, respectively. As can be noted, the proposed approach scales linearly with the size of the model, despite the element type in use: the elapsed time is linearly proportional to the number of inserted cohesive elements. These results demonstrate that the insertion of a cohesive element is based on local topological procedures, thus its performance is independent of the size of the model.

8 Fracture simulations

In this section, we demonstrate the application of TopS in dynamic fracture analysis through two finite element simulations. The first problem is a 2D simulation of mixed-mode dynamic crack growth in a pre-cracked steel plate subjected to impact loading. The second problem is a 3D simulation of mixed-mode dynamic crack growth in a pre-notched three-point-bending concrete beam.

The CZM employed in this section follows that proposed by Pandolfi and Ortiz [15], which is based on effective quantities (both tractions and displacements). The traction–separation relations for pure normal and tangential separation can display linear softening and linear unloading/reloading. This model was used by Zhang et al. [6] to investigate dynamic microbranching phenomena. Essentially, the cohesive force that resists the opening and sliding of the new surface is assumed to weaken irreversibly with increasing interface separation. Irreversibility is

[1] For reference, the tests were run on a dual-processor AMD Opteron 248 (2 × 2,193.78 MHz, 64 bits, and 16 Gb RAM).

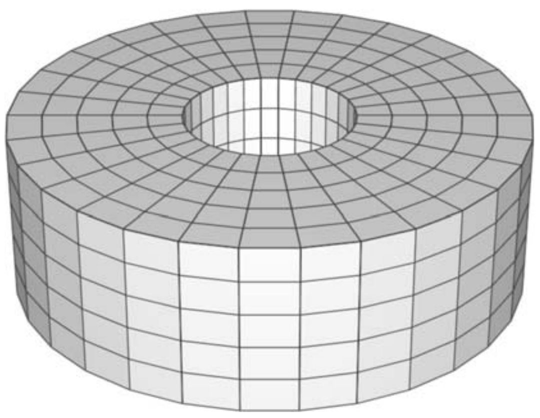


Fig. 18 General hexahedral mesh (for both Hexa8 and Hexa20) displaying a discretization of 5 × 30 × 5

retained by keeping track of the maximum displacement in the simulation history and by using it as the indicator for loading or unloading, as shown schematically in Fig. 1. Under loading condition, when the current effective opening displacement is larger than that in the history, the cohesive traction ramps down as displacement jump increases, and reduces to zero as opening reaches critical opening displacement. Decohesion is complete at this point and cohesive force vanishes thereafter (Fig. 1). If the interface reopens, the reloading path follows the unloading path in the reverse direction until the maximum effective displacement is reached, and then follows the original ramp–down relation.

The same model is employed to simulate both 2D and 3D problems in the present work. Since the two examples are presented for illustration purposes, we will focus on the ability of the data structure in handling the changing geometry. Neither the results presented nor the parameter chosen are for validation purposes.

8.1 2D mixed-mode dynamic crack growth

Kalthoff and Winkler [38] tested a plate with two edge notches subjected to impact by a projectile, as shown in Fig. 24a. The experiments demonstrated different fracture/damage behaviors of a maraging steel material under various loading rates. The material properties are listed in Table 2. The parameters are defined as following: E, ν, and ρ denote Young's modulus, Poisson's ratio, and mass density of the bulk material; parameters G_{IC} and G_{IIC} denote fracture energy for opening crack (mode-I fracture) and sliding crack (mode-II fracture); $T_n^{\max}$ and $T_t^{\max}$ are cohesive strength along normal and sliding directions, while δ_n and δ_t are the corresponding separations when material failure occurs under pure mode-I and mode-II fracture.

In this study, our objective is to simulate the brittle failure mode and investigate the overall crack propagation behavior.

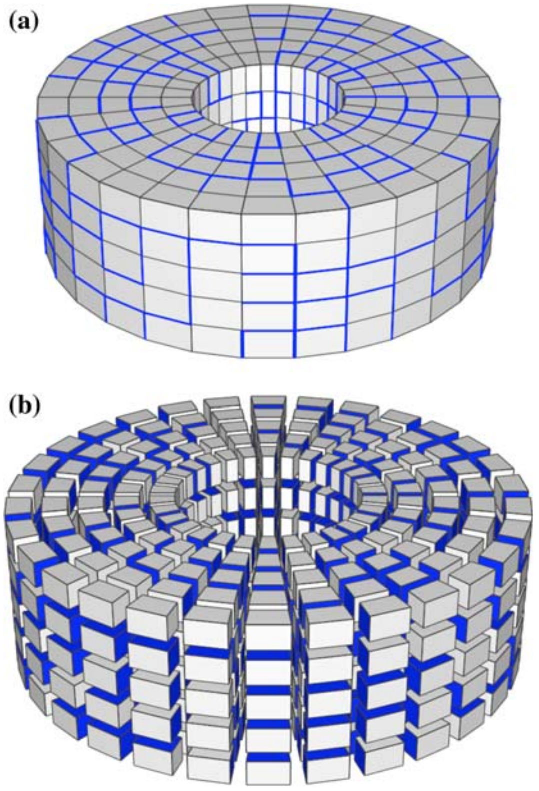


Fig. 19 Illustrative hexahedral configuration after inserting cohesive elements (*in blue*) at 20% of the facets, exemplifying the resulting (arbitrary) crack patterns: **a** external boundary; **b** model achieved after imposing a small separation in-between each bulk element interface to better illustrate the mesh with the inserted cohesive elements

The impact-loading rate is chosen as 16.5 m/s. Since the problem possesses symmetry, only half of the geometry (100×100 mm^2) is modeled, as shown in Fig. 24b.

Numerical simulation is carried out using a mesh of 80×80 with each square divided into four T6 elements. Time step is set to $\Delta t = 5 \times 10^{-3}$ μs. Figure 25 shows a set of stress contour plots taken at different times. Crack initiates at around $t = 25$ μs.

The impact load is applied along the left boundary of the lower plate section (below the initial crack plane). It creates compressive waves propagating toward the right surface within the lower section of the plate. Before the first tide of stress waves reaches the initial crack tip ($x = 50$ mm, $y = 25$ mm) at around $t = 18$ μs, the upper plate section ($y > 25$ mm) remains stress-free. When the wave reaches the crack tip, the upper crack surface, near the crack tip, stays stationary, while the lower crack surface near the crack tip is under the influence of a rightward compressive wave. This creates a tearing effect at the crack tip. Afterwards, the waves continue to propagate rightwards in the lower plate section as compressive wave, and also propagate around the crack tip into the upper section (above the initial crack plane) of the plate. The stress waves along the

Table 1 Elapsed times (in seconds) for inserting cohesive elements at all the facets of the models

Element type	Mesh discretization	Number of bulk elems	Initial number of nodes	Number of CZM elems	Final number of nodes	Time (s)
T3	100 × 600	240,000	120,600	359,400	720,000	7.636
	200 × 1,200	960,000	481,200	1,438,800	2,880,000	30.172
	300 × 1,800	2,160,000	1,081,800	3,238,200	6,480,000	67.66
	400 × 2,400	3,840,000	1,922,400	5,757,600	11,520,000	120.054
	500 × 3,000	6,000,000	3,003,000	8,997,000	18,000,000	184.782
T6	100 × 600	240,000	481,200	359,400	1,440,000	9.29
	200 × 1,200	960,000	1,922,400	1,438,800	5,760,000	36.946
	300 × 1,800	2,160,000	4,323,600	3,238,200	12,960,000	84.94
	400 × 2,400	3,840,000	7,684,800	5,757,600	23,040,000	150.04
	500 × 3,000	6,000,000	12,006,000	8,997,000	36,000,000	236.814
Q4	100 × 600	60,000	60,600	119,400	240,000	2.028
	200 × 1,200	240,000	241,200	478,800	960,000	8.126
	300 × 1,800	540,000	541,800	1,078,200	2,160,000	18.384
	400 × 2,400	960,000	962,400	1,917,600	3,840,000	32.552
	500 × 3,000	1,500,000	1,503,000	2,997,000	6,000,000	51.866
Q8	100 × 600	60,000	181,200	119,400	480,000	2.778
	200 × 1,200	240,000	722,400	478,800	1,920,000	10.996
	300 × 1,800	540,000	1,623,600	1,078,200	4,320,000	25.25
	400 × 2,400	960,000	2,884,800	1,917,600	7,680,000	43.798
	500 × 3,000	1,500,000	4,506,000	2,997,000	12,000,000	67.508
Tetra4	10 × 60 × 10	36,000	7,260	69,600	144,000	7.428
	20 × 120 × 20	288,000	52,920	566,400	1,152,000	60.972
	30 × 180 × 30	972,000	172,980	1,922,400	3,888,000	209.726
	40 × 240 × 40	2,304,000	403,440	4,569,600	9,216,000	489.204
	50 × 300 × 50	4,500,000	780,300	8,940,000	18,000,000	954.472
Tetra10	10 × 60 × 10	36,000	52,920	69,600	360,000	8.208
	20 × 120 × 20	288,000	403,440	566,400	2,880,000	65.782
	30 × 180 × 30	972,000	1,339,560	1,922,400	9,720,000	223.848
	40 × 240 × 40	2,304,000	3,149,280	4,569,600	23,040,000	545.292
	50 × 300 × 50	4,500,000	6,120,600	8,940,000	45,000,000	1,067.52
Hexa8	10 × 60 × 10	6,000	7,260	16,800	48,000	1.152
	20 × 120 × 20	48,000	52,920	139,200	384,000	9.664
	30 × 180 × 30	162,000	172,980	475,200	1,296,000	33.69
	40 × 240 × 40	384,000	403,440	1,132,800	3,072,000	81.918
	50 × 300 × 50	750,000	780,300	2,220,000	6,000,000	158.35
Hexa20	10 × 60 × 10	6,000	27,720	16,800	120,000	1.444
	20 × 120 × 20	48,000	206,640	139,200	960,000	12.336
	30 × 180 × 30	162,000	680,760	475,200	3,240,000	42.656
	40 × 240 × 40	384,000	1,594,080	1,132,800	7,680,000	101.076
	50 × 300 × 50	750,000	3,090,600	2,220,000	15,000,000	198.356

Each time reported is the average of five simulations for each specific model, inserting the elements in different random orders

upper crack surface are now tensile propagating toward the left boundary. Therefore the upper and lower surfaces of the crack are subjected to influence of stresses of opposite sign and direction along the Cartesian x coordinate (horizontal), and a strong tearing effect is created at the crack tip. When the local stress reaches the cohesive strength, a cohesive element is inserted and crack initiates. This occurs at around 25 μs. Reflective wave from the right boundary also influences the crack propagation and the crack finally grows along a direction of about 60°.

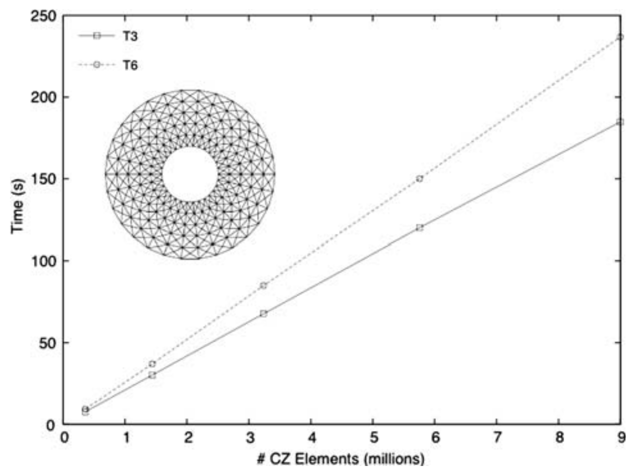


Fig. 20 Time versus number of inserted CZ elements for linear and quadratic triangular meshes

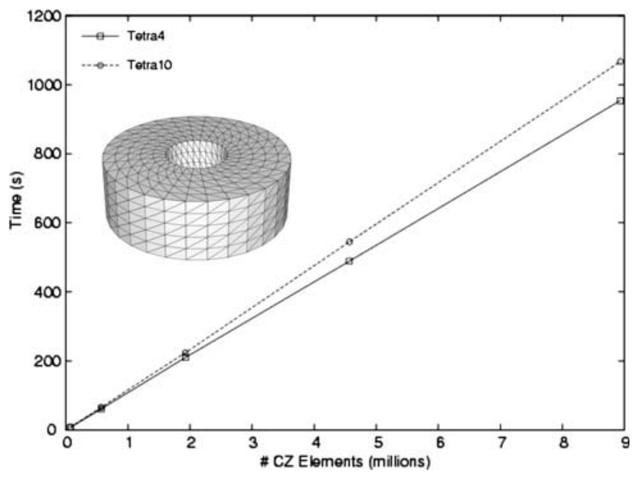


Fig. 22 Time versus number of inserted CZ elements for linear and quadratic tetrahedral meshes

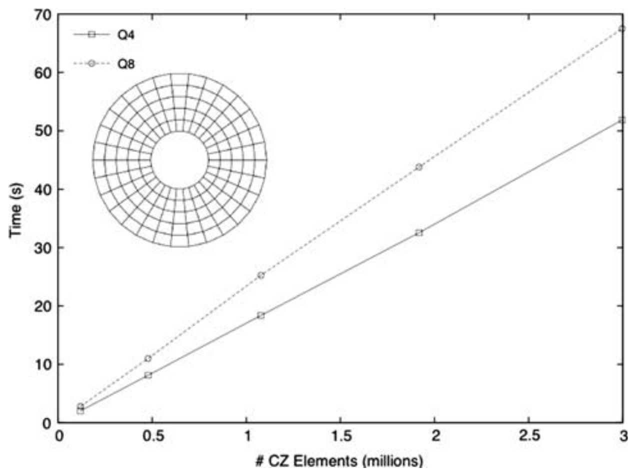


Fig. 21 Time versus number of inserted CZ elements for linear and quadratic quadrilateral meshes

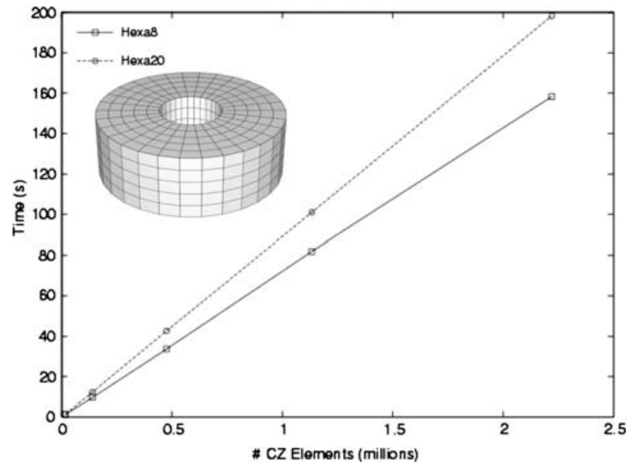


Fig. 23 Time versus number of inserted CZ elements for linear and quadratic hexahedral meshes

Due to bending effect, crack initiation also occurs at the right edge (Fig. 25d). Belytschko et al. [39] studied the same problem using extended FEM with loss of hyperbolicity criterion, and the damage zone at the bending side is also present in their analysis. We also compare the results with those obtained by Zhang and Paulino [5] using intrinsic CZMs. The overall crack behavior is similar for the two studies, while the bending crack is not as noticeable in the intrinsic CZM study because it employs a higher cohesive strength value.

Microcracks emanating from the main crack are also observed in the crack growth pattern (Fig. 25). These cracks typically arrest shortly after the main crack tip advances. We define cohesive element decohesion when all its Gauss points separation jump exceed a critical separation value. Thus, by extracting only the cohesive elements that have undergone complete decohesion, we obtain a "clean" fracture pattern, as shown in Fig. 26a. Clearly, the microcracks observed in Fig. 25 do not grow beyond one element length.

Crack velocity is also computed from crack length versus time increment information, as shown in Fig. 26b along with the Rayleigh wave speed (2,799 m/s) for comparison purpose. The average crack speed is around 1,700 m/s, which is slightly lower than that estimated by Belytschko et al. [39], which is 1,800 m/s.

8.2 3D mixed-mode dynamic crack growth

John and Shah [40] tested pre-cracked three-point-bend beam (TPB) concrete material specimens as shown in Fig. 27. Position of the pre-crack was shifted systematically to test a series of asymmetric loading conditions that result in mixed-mode fracture. Dynamic loading was applied with a modified Charpy test.

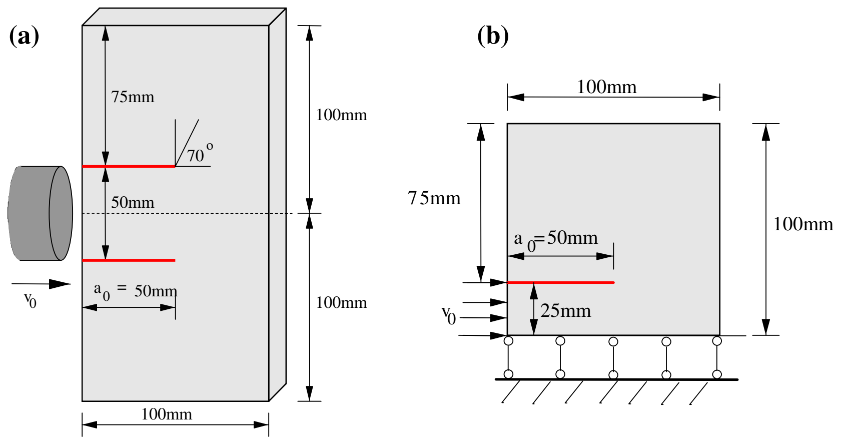


Fig. 24 **a** Geometry and loading of the Kalthoff–Winkler experiments [39]; **b** 2D plane-strain FEM simulation model

Table 2 Material properties of 18Ni(300) steel and cohesive model parameters

E (GPa)	v	ρ (kg/m^3)	$G_{Ic} = G_{IIc}$ (kJ/m^2)	$T_n^{max} = T_t^{max}$ (GPa)	$\delta_n = \delta_t$ (μm)
190	0.3	8,000	22.2	1.2	37.0

By increasing its offset (represented by the parameter α where $0 \le \alpha \le 1$), the pre-crack was subjected to mode-II fracture conditions, and the trajectory of the crack varied accordingly. Two competing crack mechanisms drove fracture initiation and growth in the specimen illustrated in Fig. 27: the crack growth initiated from the pre-crack; and the nucleation of crack at the lower center cross-section and its subsequent growth. The experiment indicates that for $\alpha < 0.77$, crack pattern is dominated by the growth of pre-crack, while for $\alpha > 0.77$ crack initiation and growth of center crack become dominant and shields the growth of pre-crack.

In this paper, we present one set of results that illustrate the capability of the data structure in analyzing the 3D fracture problem. A typical mesh for $\alpha = 0.5$ is shown in

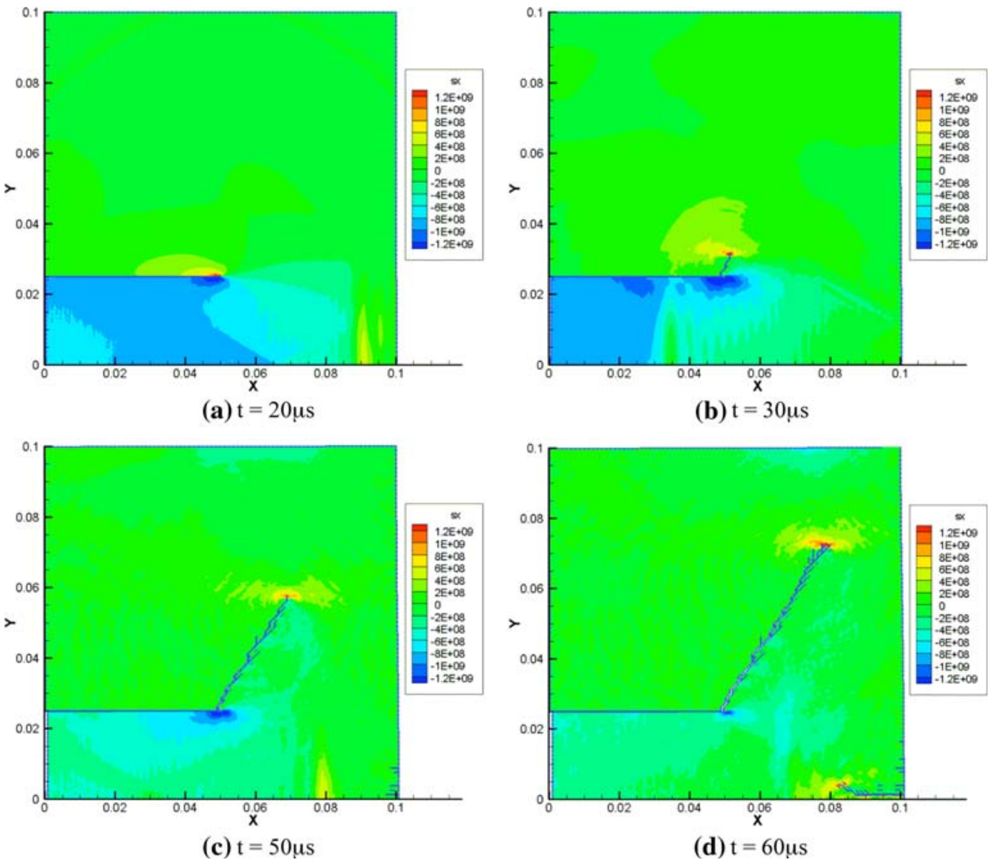
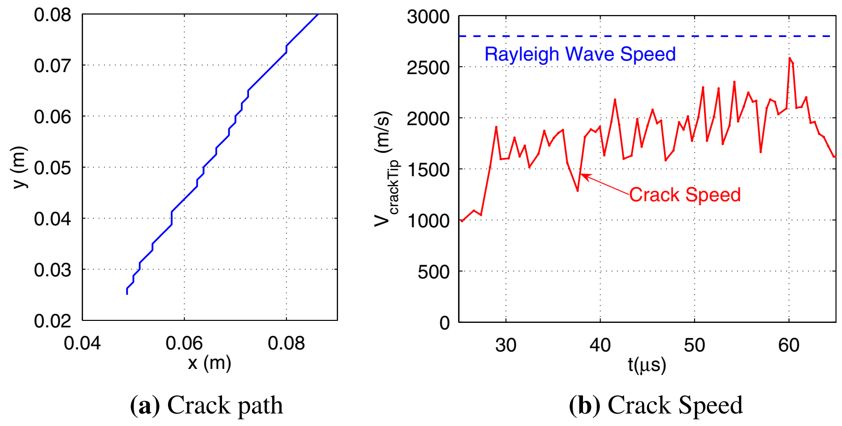


Fig. 25 Stress contour and crack propagation plots of the Kalthoff–Winkler experiment at different time instants: **a** 20 μs; **b** 30μs; **c** 50μs; **d** 60μs

Fig. 26 **a** Main crack path extracted from completely debonded cohesive surfaces in Fig. 25; **b** numerical crack speed derived directly (i.e. without post-processing) from crack length growth versus time increment. Rayleigh wave speed is 2,799 m/s



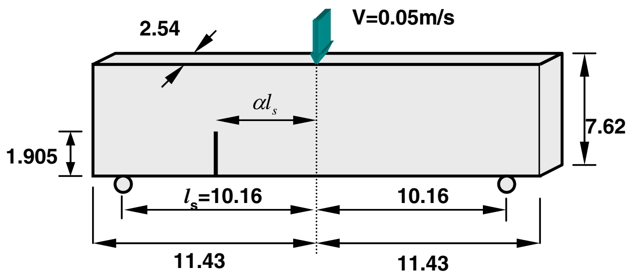


Fig. 27 Specimen dimensions (in cm) for the three-point-bending fracture test. The pre-crack is shifted to a position αl_s, where l_s is half the distance between the two supports, and $0 \leq \alpha \leq 1$

Fig. 28. Due to symmetry condition along the Cartesian y (thickness) direction, half of the geometry is used for simulation with symmetric condition appropriately applied. Mesh generation considers denser mesh within the region of projected crack growth, which is between the pre-crack and the central region. The mesh consists of 176,514 Tetra4 elements and 37,923 nodes.

Table 3 lists the material properties employed in the simulation. The notation is the same as that employed for the previous example. The tension–shear coupling parameter β, which assigns different weight for normal and sliding traction–separation in computing effective quantities [15], is set to be 2.75. Figure 28 reports principal stress and crack growth at different time instants. Before crack growth starts (Fig. 29a), stress concentration occurs both around the crack tip and along the bending side of the beam center. Crack initiates from the existing crack tip (Fig. 29 b), and afterwards the stress concentration along the central beam is gradually relieved (Fig. 29c, d). Numerous fragmentations also occur around the main crack path, while the main crack propagating path is clearly visible at an angle of around 30°.

The cohesive strength is rather low for concrete material, and thus the estimated cohesive zone size is relatively large. Based on such assumption, Ruiz et al. [41] simulated this problem using a coarse mesh. Their results did not capture the transitional α value well. For example, mid-span tensile crack begins to grow for α as low as 0.5, and becomes dominant for $\alpha = 0.6$. Ruiz et al. [41] suggested that the discrepancy between their simulation results and the experimental observation in terms of transitional α value may be contributed to choice of β value. However, our investigation indicates that the results are more sensitive to

Fig. 28 Mesh and crack pattern for pre-cracked three-point-bending specimen with crack position parameter $\alpha = 0.5$. Crack initiates from the pre-crack and grows afterwards. Mesh consists of 176,514 Tet4 elements and 37,923 nodes

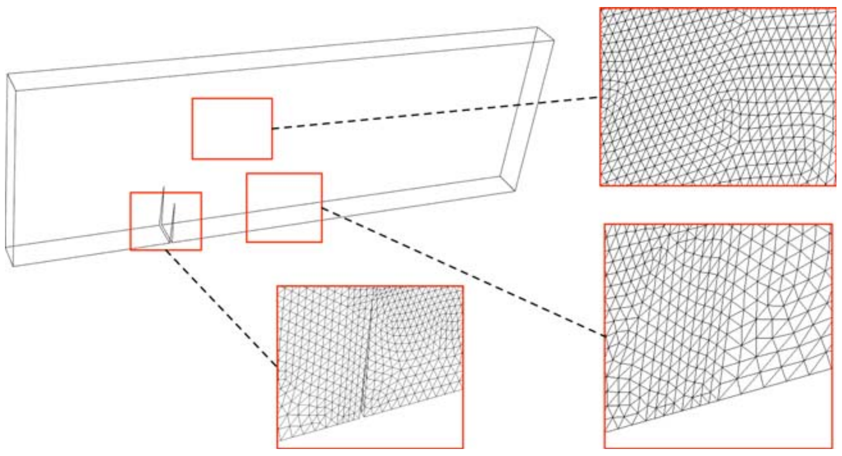


Table 3 Material properties of concrete beam specimen and cohesive model parameters

E (GPa)	ν	ρ (kg/m^3)	$G_{\mathrm{Ic}} = G_{\mathrm{IIc}}$ (J/m^2)	$T_n^{\max}$ (MPa)	$T_t^{\max}$ (MPa)	δ_n (δm)	δ_t (δm)
28	0.2	2,400	22	8	22	5.5	2.0

mesh quality rather than β for the problem. For the problem under investigation, a careful mesh quality study must precede any conclusive simulation. For 3D simulation, unstructured mesh with through-thickness size control provides better results compared to structured mesh. A detailed report for mesh quality control and simulation results is presently under development by the authors.

9 Conclusion

In this paper, we propose a topological framework for supporting both intrinsic and extrinsic fragmentation simulations. The proposed framework is general in the sense that it supports any finite element mesh with elements defined by ordered list of nodes. Based on TopS, a reduced topological data structure [17, 18], we propose a new

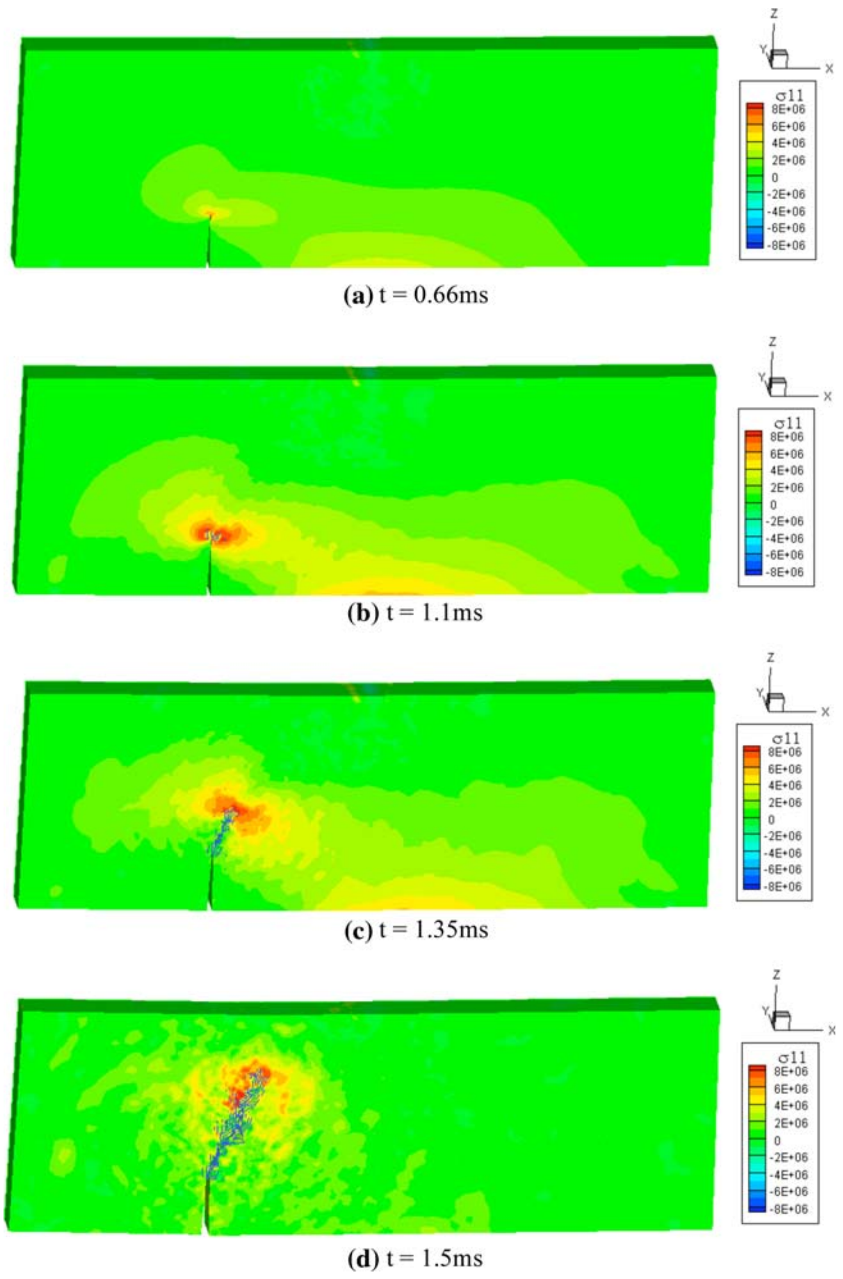


Fig. 29 Stress contour and crack pattern for pre-cracked three-point-bending specimen with crack position parameter $\alpha = 0.5$: **a** $t = 0.66$ ms; **b** $t = 1.1$ ms; **c** $t = 1.35$ ms; **d** $t = 1.5$ ms

systematic topological classification of fractured facets, thus achieving a general algorithm to insert cohesive elements along facets of bulk-elements: the same algorithm works for 2D and 3D models, including both linear and quadratic elements. The proposed approach to classify fractured facets can also be applied to other topological data structures, as long as they are complete.

We have run a set of computational experiments that demonstrate the scalability and correctness of the proposed approach. The insertion of cohesive elements is based on local topological operations. As a consequence, the time needed to insert cohesive elements at all facets of a model is linearly proportional to the number of inserted elements. Such linear scaling is demonstrated by the plots of Figures 20 through 23, including millions of CZ elements. We also have integrated TopS with actual extrinsic fragmentation simulation. Simulation results, for both 2D and 3D models, demonstrate the effective use of the proposed topological framework.

Although we have emphasized the use of TopS to provide support for fragmentation simulations with cohesive elements, it has other potential applications. For instance, TopS can be used to generate multiple arbitrary cracks (e.g., in the context of linear elastic fracture mechanics [3]) in existing crack-free meshes for complex 3D domains.

Acknowledgments Paulino gratefully acknowledges the support from NASA-Ames, Engineering for Complex Systems Program, and the NASA-Ames Chief Engineer (Dr. Tina Panontin) through grant NAG 2-1424. Both Celes and Espinha would like to thank the Tecgraf laboratory at PUC-Rio, which is mainly funded by the Brazilian oil company, Petrobras.

References

1. Cox BN, Gao H, Gross D, Rittel D (2005) Modern topics and challenges in dynamic fracture. J Mech Phys Solids 53:565–596
2. Camacho GT, Ortiz M (1996) Computational modeling of impact damage in brittle materials. Int J Solids Struct 33:2899–2938
3. Anderson TL (2005) Fracture mechanics: fundamentals and applications, 3rd edn. CRC, Boca Raton, Florida
4. Belytschko T, Chen H, Xu J, Zi G (2003) Dynamic crack propagation based on loss of hyperbolicity and a new discontinuous enrichment. Int J Numer Methods Eng 58:1873–1905
5. Zhang Z, Paulino GH (2005) Cohesive zone modeling of dynamic failure in homogeneous and functionally graded materials. Int J Plast 21:1195–1254
6. Zhang Z, Paulino GH, Celes W (2007) Extrinsic cohesive modeling of dynamic fracture and microbranching instability in brittle materials. Int J Numer Methods Eng (in press)
7. Beall MW, Shephard MS (1997) A general topology-based mesh data structure. Int J Numer Methods Eng 40:1573–1596
8. Garimella RV (2002) Mesh data structure selection for mesh generation and FEA applications. Int J Numer Methods Eng 55:451–478
9. Remacle J-F, Karamete BK, Shephard MS (2000) Algorithm oriented mesh database. In: Proceedings of the 9th international meshing roundtable, Sandia National Laboratories, pp 349–359, October
10. Remacle J-F, Shephard MS (2003) An algorithm oriented mesh database. Int J Numer Methods Eng 58:349–374
11. Owen SJ, Shephard MS (2003) Editorial: special issue on trends in unstructured mesh generation. Int J Numer Methods Eng 58:159–160
12. Löhner R (1988) Some useful data structures for the generation of unstructured grids. Commun Appl Numer Methods 4:123–135
13. Carey GF, Sharma M, Wang KC (1988) A class of data structures for 2-d and 3-d adaptive mesh refinement. Int J Numer Methods Eng 26:2607–2622
14. Hawken DM, Townsend P, Webster MF (1992) The use of dynamic data structures in finite element applications. Int J Numer Methods Eng 33(9):1795–1811
15. Pandolfi A, Ortiz M (1998) Solid modeling aspects of three-dimensional fragmentation. Eng Comput 14:287–308
16. Pandolfi A, Ortiz M (2002) An efficient adaptive procedure for three-dimensional fragmentation simulations. Eng Comput 18:148–159
17. Celes W, Paulino GH, Espinha R (2005) A compact adjacency-based topological data structure for finite element mesh representation. Int J Numer Methods Eng 64(11):1529–1565
18. Celes W, Paulino GH, Espinha R (2005) Efficient handling of implicit entities in reduced mesh representations. J Comput Inf Sci Eng Spec Issue Mesh-Based Geometric Data Process 5(4):348–359
19. Klein PA, Foulk JW, Chen EP, Wimmer SA, Gao H (2001) Physics-based modeling of brittle fracture: cohesive formulations and the applications of meshfree methods. Sandia National Laboratory, Technical Report, SAND 2001-8009
20. Ortiz M, Pandolfi A (2003) Finite-deformation irreversible cohesive elements for three-dimensional crack-propagation analysis. Int J Numer Methods Eng 44:1267–1282
21. Cirak F, Ortiz M, Pandolfi A (2005) A cohesive approach to thin-shell fracture and fragmentation. Comput Methods Appl Mech Eng 194:2604–2618
22. Song SH, Paulino GH, Buttlar WG (2007a) A bilinear cohesive zone model tailored for fracture of asphalt concrete considering viscoelastic bulk material. Eng Fract Mech 73(18):2829–2848
23. Song SH, Paulino GH, Buttlar WG (2007b) Simulation of crack propagation in asphalt concrete using an intrinsic cohesive zone model. ASCE J Eng Mech 132(11):1215–1223
24. Espinosa HD, Zavattieri PD (2003) A grain level model for the study of failure initiation and evolution in polycrystalline brittle materials. Part 1 theory and numerical implementation. J Mater 35:333–364
25. Zavattieri P, Espinosa HD (2001) Grain level model analysis of crack initiation and propagation in brittle materials. Acta Mater 49:4291–4311
26. Xu X, Needleman A (1995) Numerical simulations of dynamic crack growth along an interface. Int J Fract 74:289–324
27. Pandolfi A, Krysl P, Ortiz M (1999) Finite element simulation of ring expansion and fragmentation: the capturing of length and time scales through cohesive models of fracture. Int J Fract 95:279–297
28. Matous K, Geubelle PH (2006) Multiscale modelling of particle debonding in reinforced elastomers subjected to finite deformations. Int J Numer Methods Eng 65:190–223
29. Iesulauro E, Ingraffea AR, Arwade S, Wawrzynek PA (2002) Simulation of grain boundary decohesion and crack initiation in aluminum microstructure models. In: Reuter WG, Piascik RS (eds) Fatigue and fracture mechanics: 33rd volume, ASTM STP 1417. American Society for Testing and Materials, West Conshohocken, pp 715–728

30. Weiler K (1986) Topological structures for geometric modeling, Ph.D. Thesis, Rensselaer Polytechnic Institute, New York
31. Weiler K (1988) The radial edge structure: a topological representation for non-manifold geometric boundary modeling. In: Wozney M, McLaughlin H, Encarnacao J (eds) Geometric Modeling for CAD Applications, North-Holland, pp 3–36
32. Mäntylä M (1988) An introduction to solid modeling. Computer Science Press, Rockville
33. Wawrzynek PA, Ingraffea AR (1987) Interactive finite element analysis of fracture processes: an integrated approach. Theor Appl Fract Mech 8:137–150
34. Martha LF, Wawrzynek PA, Ingraffea AR (1993) Arbitrary crack representation using solid modeling. Eng Comput 9:63–82
35. Paulino GH, Jin Z-H, Dodds RH Jr (2003) Failure of functionally graded materials. In: Karihaloo B, Knauss WG (eds) Comprehensive structural integrity, vol 2, Chapter 13. Elsevier Science, Amsterdam, pp 607–644
36. Jin Z, Paulino GH, Dodds RH Jr (2003) Cohesive fracture modeling of elastic–plastic crack growth in functionally graded materials. Eng Fract Mech 70:1885–1912
37. De Floriani L, Hui A (2003) A scalable data structure for three-dimensional non-manifold objects. In: Proceedings of the Eurographics symposium on geometry processing, Aachen, Germany, pp 72–82
38. Kalthoff JF, Winkler S (1987) Failure mode transition at high rates of shear loading. In: Chiem CY, Kunze HD, Meyer LW (eds) Proceedings of the international conference on impact loading and dynamic behavior of materials, Bremen, Germany, pp 185–195
39. Belytschko T, Chen H, Xu J, Zi G (2003) Dynamic crack propagation based on loss of hyperbolicity and a new discontinuous enrichment. Int J Numer Methods Eng 58:1873–1905
40. John R, Shah SP (1990) Mixed-mode fracture of concrete subjected to impact loading. J Struct Eng 116:585–602
41. Ruiz G, Pandolfi A, Ortiz M (2001) Three-dimensional cohesive modeling of dynamic mixed-mode fracture. Int J Numer Methods Eng 52:97–120