Representing Simplicial Complexes with Mangroves

David Canino and Leila De Floriani

Department of Computer Science, Bioengineering, Robotics and Systems Engineering (DIBRIS), University of Genova, Genova, Italy
{canino,deflo}@disi.unige.it

Summary. Simplicial complexes are extensively used for discretizing digital shapes in two, three, and higher dimensions within a variety of application domains. There have been many proposals of topological data structures, which represent the connectivity information among simplices. We introduce the *Mangrove Topological Data Structure (Mangrove TDS)* framework, a tool which supports the efficient implementation of data structures for simplicial complexes of any dimension under the same application interface. Our framework is based on a graph-based representation of connectivity relations, that we call the *mangrove*. It can be customized in order to simulate the content of any topological data structure with a negligible overhead. Thus, the Mangrove TDS framework is *extensible*, and supports the most diverse modeling needs. We also provide implicit representations of those simplices, which are not directly encoded in a specific topological data structure. Our tests show that these representations, that we call *ghost simplices*, improve the expressive power and the efficiency of topological queries. In order to prove the validity of our approach, we design two topological data structures, specific for non-manifold complexes, within our framework. We perform comparisons with some widely-used representations in the literature as well as with libraries available in the public domain.

1 Introduction

Simplicial complexes are used to discretize digital shapes in many applications, including computer graphics, solid modeling, numerical simulations, scientific visualization, and geographic data processing. Many tools handle *manifold* shapes, i.e., subsets of the Euclidean space such that each neighborhood of every point is homeomorphic to a ball. Objects, which violate this property, are *non-manifold*, and arise in the finite element simulations [25] and in the topological analysis of data in high dimensions [3, 5]. Many topological data structures have been developed for retrieving connectivity information of simplicial complexes [18]. Most of representations are specific for shape optimization [7], geometry processing [6, 32], and numerical simulations [1, 25].

A framework, which supports topological data structures under a common interface is lacking in the literature [48]. It should be the basis for performing quantitative comparisons, regarding their performances. Topological data structures are implemented in heterogenous libraries, whose efficiency depends not only on their intrinsic properties, but also on the programmers' technical ability and on their design. This framework should also be a common platform for designing and simulating topological data structures efficiently. The key idea consists of creating a prototype of a topological data structure, which is customized in order to satisfy any modeling need in the same spirit of the *rapid prototyping* techniques in the manufacturing [52]. Hence, this framework may be *extended* with new data structures, and the internal representation of a complex may be dynamically replaced in order to choose the most *efficient* one for a specific task. Thus, it would be different from many tools in the literature [6,7,14], since they are defined on a specific representation, which cannot be replaced dynamically at run-time. Actually, some tools can be modified dynamically [23, 34, 45, 47], but only in a predefined way and without messing with their internal representation, which remains almost unchanged.

The connectivity information of simplices, formalized by *topological relations*, can be described as a directed graph, which we call the *mangrove*, whose nodes and arcs correspond, respectively, to simplices and topological relations, encoded in any topological data structure. In this paper, we propose the *Mangrove Topological Data Structure (Mangrove TDS)* framework, which is geared to the design of topological data structures, represented by mangroves, under the same application interface.

Another contribution of this work is provided by what we call the *ghost simplices*. *Adjacency-based* data structures, e.g., the *Indexed data structure with Adjacencies* [44], encode only vertices, and top simplices (those that do not bound other simplices). They are more compact than the *incidence-based* ones, in which all simplices are encoded explicitly [18]. However, it may be necessary to execute topological queries also on those simplices, which are not encoded explicitly. Thus, it is mandatory to define an *implicit representation* for these simplices in order to execute topological queries on complexes of any dimension and with a domain not necessarily manifold [23]. Most of the implicit representations in the literature are defined only for 2D and 3D manifolds [46]. There exist also implicit representations [8, 12], which are suitable for non-manifolds, but they result in a expensive representation, if used in high dimensions. Here, we propose a new implicit representation of a simplex (that we call the *ghost simplex*), which is always formed by four indices, despite the dimension and the number of top simplices in the complex. We show that ghost simplices improve the efficiency of topological queries and the expressive power of adjacency-based representations [10].

We have designed many data structures [11, 16, 17, 19, 20, 24] in order to evaluate the modeling capabilities of our framework [10]. Their implementations, specific for simplicial complexes, are contained in the *Mangrove TDS Library* [38]. We have evaluated the efficiency of topological queries on these

data structures with respect to popular representations in the literature as well as with libraries in the public domain. Our tests show that the *Incidence Simplicial (IS)* [19] data structure and the *Generalized Indexed data structure with Adjacencies (IA*)* [11] are the most efficient ones in this group [10].

The remainder of this paper is organized as follows. In Sect. 2, we summarize background notions, and, in Sect. 3, we review related work. In Sect. 4, we introduce our Mangrove TDS framework, and, in Sect. 5, we propose the implementations of the IS and the IA* data structures within our framework. In Sect. 6, we propose and introduce ghost simplices, while, in Sect. 7, we show quantitative results regarding our implementations. Finally, in Sect. 8, we draw some concluding remarks and discuss future developments.

2 Background Notions

A Euclidean *simplex* σ of *dimension* $k = dim(\sigma)$ is the linear combination of $k+1$ points in the Euclidean space $\mathbb{E}^n$, with $0 \leq k \leq n$. Any Euclidean d-simplex σ', with $0 < d \leq k$, generated by a subset of $d+1$ vertices of σ, is a *d-face* of σ. We say that σ is a k-face of itself. The number of d-faces of σ is $\binom{k+1}{d+1}$ [24]. A *simplicial d-complex* Σ is a finite collection of simplices of dimension at most d if (i) faces of each simplex belong to Σ, and (ii) for any simplices σ and σ', either $\sigma \cap \sigma' = \emptyset$ or $\sigma \cap \sigma'$ is a common face. A d-simplex in Σ is *maximal*. The *domain* of Σ is the union as point sets of its simplices.

The *combinatorial boundary* $B(\sigma)$ of any k-simplex σ contains its faces. The *star* of simplex σ, denoted as $St(\sigma)$, contains simplices σ' such that σ belongs to $B(\sigma')$. Simplices such that their star is empty are *top simplices*. A *non-regular* simplicial d-complex Σ contains top simplices, which are not maximal. Two k-simplices are *adjacent* if they share a $(k-1)$-simplex, while any two vertices are adjacent if they are connected by a common 1-simplex. The *link* of any simplex σ, denoted as $Lk(\sigma)$, is formed by all the faces of simplices in $St(\sigma)$, such that σ does not belong to their boundaries.

A *k-path* is a sequence of $(k+1)$-simplices in any simplicial complex Σ such that two consecutive simplices are adjacent. Two simplices σ and σ' are *k-connected* if they are connected by a k-path. A subset Σ' of Σ, formed by k-connected simplices, is a *k-connected subcomplex* of Σ. Any maximal $(k-1)$-connected subcomplex of Σ, formed only by top k-simplices, is a *k-cluster*.

A k-simplex σ in a simplicial d-complex Σ is *manifold* if $Lk(\sigma)$ is homeomorphic to the $(d-k)$-sphere, otherwise σ is a *non-manifold singularity*. If all 0-simplices of Σ are manifold, then Σ is a *combinatorial manifold* and its domain is a manifold. In any case, these tests are not always decidable [41].

3 Related Work

Many topological data structures have been proposed in the literature [1, 18, 25]. Informally, topological data structures encode the connectivity informa-

tion for any subset of simplices. They are classified with respect to the dimension, the type of entities, and the domain of the complex. A data structure for non-manifolds should be *scalable*, i.e., should exhibit a small overhead, if applied to manifolds, with respect to a data structure, specific for manifolds.

Edge-based data structures, specific for cell 2-complexes, encode oriented boundaries of 2-cells as lists of oriented edges (*half-edges*), like the *Half-Edge (HE)* data structure [39] and its extensions [6, 9, 31, 40, 48], which are used in popular libraries in computational geometry [14, 42]. The *Radial-Edge* data structure [51] is one of the first extensions of the HE data structure for non-manifold 2-complexes, but it exhibits a large overhead if applied to manifolds. The *Partial Entity* [35] and the *Q-Complex* [53] representations are compact variants of the Radial-Edge data structure. These data structures have been extended to 3-complexes by encoding the oriented 2-cells (*half-faces*) on the boundary of 3-cells [22, 37]. The *OpenVolumeMesh* data structure [32] represents non-manifold 3-complexes, and is the basis of a popular library [43]. These representations, including the *X-Maps* [13], are specializations of the *Combinatorial Maps* [36], which are based on the *cell tuples* [8].

Edge-based data structures are more verbose than the *incidence-based* representations [18]. These latter encodes all cells in a complex and a subset of their boundary and star, like the *Incidence Graph* [24], and its dimension-independent restrictions to simplicial complexes, i.e., the *Simplified Incidence Graph* [16] and our *Incidence Simplicial* [19] data structure.

Adjacency-based data structures are alternative compact representations for simplicial complexes. They encode only vertices and top simplices, which are not on the boundary of other simplices, plus their adjacent simplices. The dimension-independent *Indexed data structure with Adjacencies (IA)* [44] is limited to regular simplicial complexes of any dimension. Slight variants of this latter are implemented in [50] and in tools popular in the finite element analysis [7, 45, 49]. Two extensions of the IA data structure for non-manifold simplicial 2- and 3-complexes, embedded in the Euclidean space $\mathbb{E}^3$, are introduced in [17,20]. Our *Generalized Indexed data structure with Adjacencies* [11] extends the IA data structure to non-manifold simplicial complexes of any dimension, not necessarily embedded in the Euclidean space.

Compact representations for manifold simplicial complexes are obtained by encoding topological relations implicitly, e.g., *cell tuples* [8]. The *Corner-Table* data structure [46] and its variants [27–30, 33] are specific for manifold simplicial 2- and 3-complexes. A compact representation for regular complexes of dimension up to 3 is based on the link of vertices and edges [4]. The *Cell-Chains* [12] extend cell tuples to non-manifold complexes of any dimension.

4 The Mangrove TDS framework

As shown in [18], the connectivity information of simplices in a simplicial d-complex Σ is expressed by *topological relations*. Let Σ^k be the collection of

k-simplices in Σ (with $0 \leq k \leq d$). A *topological relation* (σ, σ') in $\Sigma^k \times \Sigma^m$ is denoted as $\sigma \sim_{km} \sigma'$. It can be: (i) a *boundary relation*, with $k > m$, if σ' is a face of σ; (ii) a *co-boundary relation*, with $k < m$, if σ' belongs to $St(\sigma)$; (iii) an *adjacency relation*, with $k = m \neq 0$, if σ and σ' share a $(k-1)$-face, or they are adjacent vertices, if $k = m = 0$. Given a k-simplex σ in Σ, the *relational operator* $R_{k,m}(\sigma) = \{\sigma' \in \Sigma^m . \sigma \sim_{km} \sigma'\}$ is confused with $\sim_{km}$.

Topological relations provide an effective framework for representing simplicial complexes by *topological data structures*, which encode explicitly only a subset of topological relations, restricted to a subset of simplices in Σ. The main contribution of our *Mangrove Topological Data Structure (Mangrove TDS)* framework consists of representing any topological data structure under a common application interface. The key idea of our approach consists of providing a *generic prototype* of a topological data structure, which can be customized in order to simulate the content of any representation in the same spirit of the *rapid prototyping* techniques in the manufacturing [52].

This prototype is obtained by describing topological relations in any simplicial d-complex Σ as a directed graph $\mathcal{G}_\Sigma = (\mathcal{N}, \mathcal{A})$, in which (i) each node n_σ corresponds to one k-simplex in Σ, and (ii) each arc (n_σ, n'_σ) in $\mathcal{A}$ connects two nodes, representing a k-simplex σ and a m-simplex σ' (with $0 \leq k, m \leq d$) such that $\sigma \sim_{km} \sigma'$. Graph $\mathcal{G}_\Sigma$ describes all simplices and topological relations in Σ, thus it cannot be used in the applications due to its high storage cost. However, any topological data structure $\mathcal{M}_\Sigma$ is described as a subgraph of $\mathcal{G}_\Sigma$, indicated as $\mathcal{G}_\Sigma^\mathcal{M}$. This latter is formed by nodes and arcs of $\mathcal{G}_\Sigma$, corresponding, respectively, to simplices and topological relations, which are directly encoded in $\mathcal{M}_\Sigma$. Arcs of graph $\mathcal{G}_\Sigma^\mathcal{M}$ are classified on the basis of the topological relations they represent as the *boundary*, *co-boundary* and *adjacency* arcs. Boundary, co-boundary and adjacency arcs, together with nodes of $\mathcal{G}_\Sigma^\mathcal{M}$, define three spanning subgraphs of $\mathcal{G}_\Sigma^\mathcal{M}$, that we call the *boundary*, *co-boundary* and *adjacency* graphs, respectively. These graphs belong to a specific class of graphs, known as *mangroves*, such that nodes and arcs, reachable from any node, form a tree [2]. In particular, graph $\mathcal{G}_\Sigma^\mathcal{M}$ is (i) a *global mangrove*, if it encodes all nodes of graph $\mathcal{G}_\Sigma$, corresponding to all simplices in Σ, (ii) a *partial mangrove*, if it encodes only some nodes of $\mathcal{G}_\Sigma$, corresponding to a subset of simplices in Σ. Thus, mangroves allow to reduce the problem of designing any topological data structure $\mathcal{M}_\Sigma$ to the representation of graph $\mathcal{G}_\Sigma^\mathcal{M}$, regardless what simplices, topological relations, and which domain are described.

Generic Encoding of Connectivity Information

A specific mangrove $\mathcal{G}_\Sigma^\mathcal{M}$, which does not encode necessarily all simplices in Σ, can be represented by an adjacency-list data structure. All nodes of graph $\mathcal{G}_\Sigma^\mathcal{M}$, which correspond to p-simplices in Σ (for each $0 \leq p \leq d$), are stored in one array, which we call the *SimplicesContainer*. This latter supports the garbage collector mechanism for reusing those locations, which are marked as *deleted* by editing operators. In order to access locations sequentially, safe iterators, which skip deleted locations automatically, are also provided. Any

location of one SimplicesContainer array contains a node n_σ of $\mathcal{G}_\Sigma^\mathcal{M}$, which corresponds to any p-simplex σ. This location is accessed in constant time through a handle (p, i), which we call the *SimplexPointer*, where i is the index of its location in the SimplicesContainer array of interest, identified by p. This location also contains nodes of $\mathcal{G}_\Sigma^\mathcal{M}$ which are adjacent to n_σ through boundary, co-boundary, and adjacency arcs in $\mathcal{G}_\Sigma^\mathcal{M}$, respectively. Note that not all types of arcs will be present in $\mathcal{G}_\Sigma^\mathcal{M}$, e.g., adjacency arcs will not be present when representing an incidence-based data structure [10]. Auxiliary information, e.g., Euclidean coordinates and field values, are dynamically allocated at run-time and associated as attributes with nodes of $\mathcal{G}_\Sigma^\mathcal{M}$ as in [31,32,48].

Any mangrove $\mathcal{G}_\Sigma^\mathcal{M}$ is a *dynamic plugin*, which satisfies a common application interface in a transparent way by using meta-programming techniques. We consider the following operations, defined on a simplex σ: (i) BOUNDARY, which retrieve all faces of σ, (ii) STAR, which retrieves simplices in the star of σ, (iii) ADJACENCY, which retrieves all simplices adjacent to σ, (iv) LINK, which retrieves the link of σ, and (v) IS-MANIFOLD, which checks whether σ is manifold. These operations are suitable for many modeling needs, and are the basis for high-level operations, e.g., those in [48].

Any topological data structure $\mathcal{M}_\Sigma$ is made available within our framework by providing its mangrove $\mathcal{G}_\Sigma^\mathcal{M}$ and the implementations of queries. This process does not require additional overhead with respect to any implementation of $\mathcal{M}_\Sigma$ outside our framework, since only the content of $\mathcal{M}_\Sigma$ is encoded.

5 Implementations of Topological Data Structures

We have designed many data structures [11,16,17,19,20,24] within our framework [10]. Here we present the *Incidence Simplicial* data structure [19] and the *Generalized Indexed data structure with Adjacencies* [11], since their complementary properties validate the modeling capabilities of our contribution.

5.1 The Incidence Simplicial data structure

The *Incidence Simplicial (IS)* data structure [19] encodes all simplices in any simplicial d-complex Σ, not necessarily embedded in the Euclidean space. For any p-simplex σ, it encodes its $(p-1)$-faces, i.e., boundary relation $\mathcal{R}_{p,p-1}(\sigma)$, and partial co-boundary relation $\mathcal{R}^*_{p,p+1}(\sigma)$, which relates σ with one arbitrary $(p+1)$-simplex in the star of σ, corresponding to each connected component in the link of σ. Fig. 1(a) shows two connected components (in bold lines) in the link of vertex v (with label 0), which are formed, respectively, by vertex v', and triangle f_t, plus edge e_f. These latter correspond, in the star of v, to top edge w and to tetrahedron t and top triangle f (represented by edge e). Thus, partial co-boundary relation $\mathcal{R}^*_{0,1}(v) = (w, e)$.

The IS data structure is represented by a global mangrove $\mathcal{G}_\Sigma^{IS}$, which we call the *IS-graph*, where any node n_σ corresponds to a simplex σ, any boundary

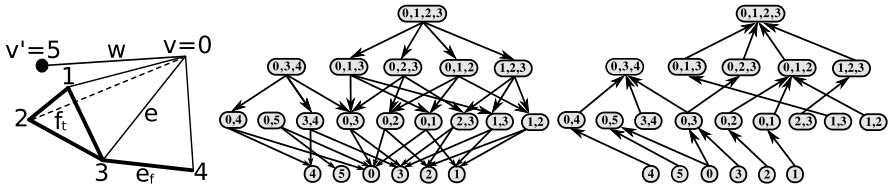


Fig. 1. (a) Two connected components (in bold lines) in the link of vertex v (with label 0) in a simplicial 3-complex. The (b) boundary and (c) co-boundary graphs in the global mangrove representing the IS data structure.

arc to a boundary relation $\mathcal{R}_{p,p-1}$, for $0 < p \leq d$, and any co-boundary arc to a partial co-boundary relation $\mathcal{R}^*_{p,p+1}$, for $0 \leq p < d$. Encoding the IS-graph $\mathcal{G}^{IS}_{\Sigma}$ does not require to store any adjacency arc in the adjacency-list data structure proposed in Sect. 4. For each node n_σ, endpoints of boundary and co-boundary arcs, outgoing from n_σ, are encoded, as shown in Figs. 1(b-c) for a simplicial 3-complex. Topological queries, like the BOUNDARY and the STAR queries, are expressed as breadth-first traversals of nodes reachable from any node n_σ in the IS-graph, in the same spirit of algorithms in [19].

5.2 The IA* data structure

The *Generalized Indexed data structure with Adjacencies (IA*)* [11] encodes all vertices and top simplices in any simplicial d-complex Σ, not necessarily embedded in the Euclidean space. For each top p-simplex σ (with $1 \leq p \leq d$), it encodes vertices of σ, i.e., boundary relation $\mathcal{R}_{p,0}(\sigma)$, all top p-simplices adjacent to σ, i.e., adjacency relation $\mathcal{R}^*_{p,p}(\sigma)$. For each vertex v, it encodes partial co-boundary relation $\mathcal{R}^*_{0,p}(v)$, for $1 \leq p \leq d$, consisting of one top p-simplex for each p-cluster in the star of v, i.e., any maximal $(p-1)$-connected subcomplex of $St(v)$, formed by top p-simplices. Fig. 2(a) shows four clusters in the star of vertex v (with label 1) in a simplicial 3-complex. Here, $\mathcal{R}^*_{0,1}(v) = ([1,2])$ contains top edge $[1,2]$, $\mathcal{R}^*_{0,2}(v) = ([1,3,4],[1,8,4]$ corresponds to two 2-clusters, in red and blue, respectively, and $\mathcal{R}^*_{0,3}(v) = [1,11,12,14]$ corresponds to a 3-cluster, in green. The IA* data structure also encodes partial co-boundary relation $\mathcal{R}^*_{p-1,p}(\tau)$, consisting of top p-simplices in the star of any $(p-1)$-simplex τ on the boundary of more than two top p-simplices. Relation $\mathcal{R}^*_{p-1,p}(\tau)$ allows for a compact encoding of $\mathcal{R}^*_{p,p}(\sigma)$ along one of its $(p-1)$-faces τ. If more than two top p-simplices are incident at τ, e.g., four triangles in the star of edge $[1,3]$ in Fig. 2(a), then $\mathcal{R}^*_{p,p}(\sigma)$ along τ is encoded as a reference to $\mathcal{R}^*_{p-1,p}(\tau)$, which is stored only once, instead of being replicated for each top p-simplex in $St(\tau)$. On the contrary, a top p-simplex adjacent to σ along τ is encoded, like top triangles $[1,8,9]$ and $[1,9,10]$ in Fig. 2(a).

The IA* data structure is represented by a partial mangrove $\mathcal{G}^{IA^*}_{\Sigma}$, which we call the *IA*-graph*, where nodes correspond to vertices and top simplices,

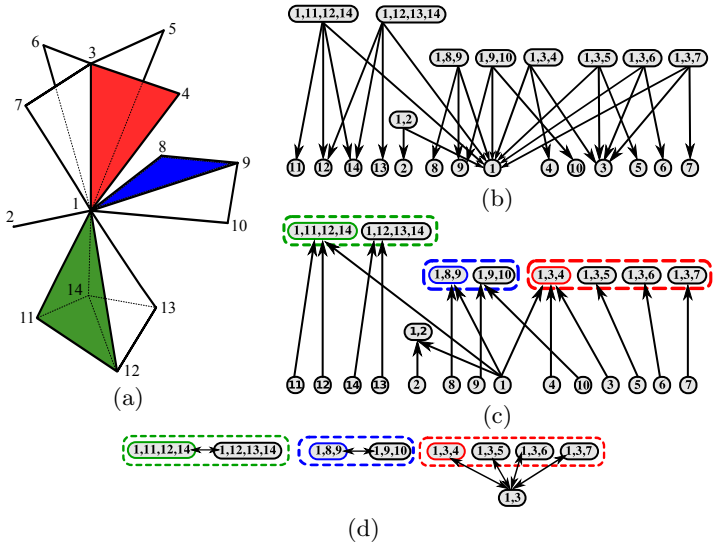


Fig. 2. (a) Four clusters in the star of vertex v (with label 1) in a simplicial 3-complex. The corresponding (b) boundary, (c) co-boundary, and (d) adjacency graphs in the partial mangrove representing the IA* data structure.

boundary arcs to relations $\mathcal{R}_{p,0}$, co-boundary arcs to relations $\mathcal{R}^*_{0,p}$, and adjacency arcs to relations $\mathcal{R}^*_{p,p}$ and $\mathcal{R}^*_{p-1,p}$. Figs 2(b-d) show, respectively, boundary, co-boundary, and adjacency graphs in the IA*-graph, describing a simplicial 3-complex. The basic operation for performing topological queries consists of retrieving top simplices in the star of a vertex v, which is expressed as a breadth-first traversal of nodes reachable from node n_v in the IA*-graph. Remaining topological queries can be implemented as discussed in [11].

6 Implicit Representations of Simplices

Adjacency-based data structures, e.g., the IA* data structure, are compact, since only vertices and top simplices are encoded explicitly [18]. In any case, only topological relations on vertices and top simplices are retrieved in optimal time, while the time complexity of algorithms retrieving other relations depends on the representation of simplices [11]. Simplices can be represented by their vertices, but this representation does not scale well to high-dimensions, and does not improve efficiency of topological queries [10]. Thus, it is necessary to represent implicitly those simplices, which are not necessarily encoded [23].

Cell tuples [8] are one of the most common implicit representations in the literature, and are defined on any manifold complex Σ. Connectivity information for simplices of Σ is described by two Hasse graphs $\mathcal{H}_b$ and $\mathcal{H}_c$ [21]. They are graph-based representations of topological relations in the incidence graph [24], i.e., for any p-simplex σ, boundary relation $\mathcal{R}_{p,p-1}(\sigma)$ (as in the

IS data structure), and co-boundary relation $\mathcal{R}_{p,p+1}(\sigma)$, formed by $(p+1)$-simplices in the star of σ. Since these graphs contain the same nodes and arcs (but with opposite orientations), we focus our attention to $\mathcal{H}_c$. Fig. 3(b) shows graph $\mathcal{H}_c$ for a 3-complex. Any cell-tuple $(\sigma_0, \ldots, \sigma_d)$ is a $(d+1)$-tuple of simplices in Σ such that $dim(\sigma_i) = i$ (with $0 \leq i \leq d$) and σ_{i+1} belongs to $St(\sigma_i)$. It corresponds to a maximal path in $\mathcal{H}_c$, which connects two nodes, describing, respectively, a vertex and a maximal simplex in its star. Cell-tuple $((4), (4,7), (4,5,7), (4,5,6,7))$ in Fig. 3(b), corresponds to a maximal path (in red) between vertex 4 and tetrahedron $(4,5,6,7)$. Cell-tuples describe the complete connectivity information of any complex [36], but they are not suitable for high-dimensional d-complexes, since they require $d+1$ references.

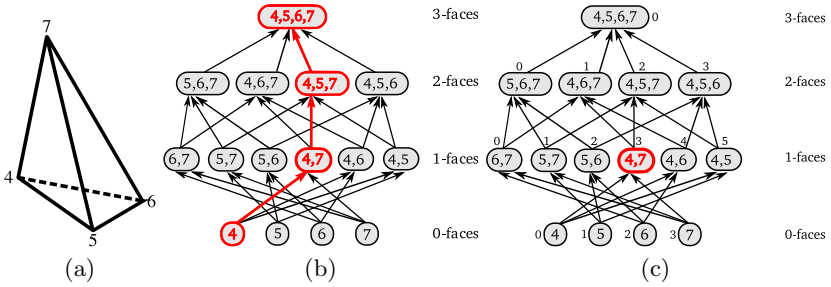


Fig. 3. (a) In any 3-simplex, (b) a *cell-tuple* is any maximal path (in red) in Hasse graph $\mathcal{H}_c$. (c) On the contrary, a *ghost simplex* corresponds only to one of its faces, e.g., edge $(4,7)$ (in red) is described by ghost simplex $(3,0,1,3)$.

In order to solve these drawbacks, we propose a new implicit representation of simplices, not necessarily encoded in any adjacency-based data structure. The key idea of our approach consists of describing implicitly a p-simplex σ (which we call the *child simplex*) in any simplicial d-complex Σ as a p-face of any top t-simplex σ' (which we call the *parent simplex*) in the star of σ, such that $0 \leq p \leq t \leq d$. Thus, simplex σ is represented by a tuple (t, i, p, j), which we call the *ghost simplex* of σ, where: (i) t is the dimension of parent simplex σ', (ii) i is the unique identifier of σ' in the collection of top t-simplices in Σ, (iii) p is the dimension of child simplex σ, (iv) j is the unique identifier of σ, seen as any p-face of σ'. Note that indices i and j depend on which rule is exploited in order to enumerate top simplices in Σ and their faces. The number of p-faces of any t-simplex is $\binom{t+1}{p+1}$ [24], then $0 \leq j \leq \binom{t+1}{p+1}$. Fig. 3(c) shows how enumerating faces of a top 3-simplex, which is identified by label 0. Any 3-simplex has 6 edges, then node of graph $\mathcal{H}_c$, corresponding to edge $[4,7]$ (in red), has identifier 3, and is represented by ghost simplex $(3,0,1,3)$. Note that any p-simplex σ is represented by several ghost simplices, one for each top simplex in its star. Thus, this representation is not unique.

Ghost simplices provide an efficient way to associate a simplex with one top simplex in its star without traversing graphs $\mathcal{H}_b$ and $\mathcal{H}_c$ explicitly. This speeds up the retrieval of topological queries in adjacency-based data structures, since they encode adjacency relations, restricted to top simplices, which can be accessed quickly thanks to ghost simplices. The key idea of our approach consists of restricting graphs $\mathcal{H}_b$ and $\mathcal{H}_c$ to all faces of a top t-simplex σ' in the star of any p-simplex σ (with $0 \leq p \leq t \leq d$), and associating a ghost simplex with each face of σ' (including σ'). This is achieved by enumerating nodes of $\mathcal{H}_b$ and $\mathcal{H}_c$ by dimension and associating their unique identifiers with each node (see Fig. 3(c)). The resulting ghost simplices are defined with respect to σ'. The dimension t and the unique identifier of σ' are known as well as the dimension p of child simplex σ. Thus, there is the need to retrieve the unique identifiers of ghost simplices of interest, seen as faces of σ'. These identifiers are the same for all top t-simplices in any simplicial d-complex Σ. Thus, they can be stored explicitly as a graph $\mathcal{H}_t$, one for each class of top t-simplices in Σ (for all $0 \leq t \leq d$) in order to speed-up the retrieval of topological queries. Each node of graph $\mathcal{H}_t$ corresponds to a p-face σ of any top t-simplex σ', and contains unique identifiers of faces of σ', which are, respectively, $(p-1)$-faces of σ (i.e., boundary relation $\mathcal{R}_{p,p-1}$ in $\mathcal{H}_b$), and $(p+1)$-simplices in the star of σ (i.e., co-boundary relation $\mathcal{R}_{p,p+1}$ in $\mathcal{H}_c$). Figs 4(a-b) show, respectively, indices of these faces (represented as ghost simplices) for any 3-simplex. In Fig. 4(a), the content of node in red is $[0, 3]$, and, if we apply these idea to 3-simplex in Fig. 3(a), then its immediate boundary relation contains ghost simplices $(3, 0, 0, 0)$ and $(3, 0, 0, 3)$. As shown in Fig. 4(b), its immediate co-boundary relation contains ghost simplices $(3, 0, 2, 1)$ and $(3, 0, 2, 2)$.

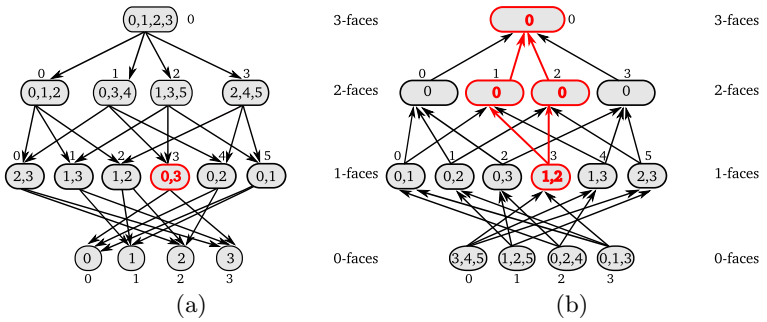


Fig. 4. Graph-based representations of immediate (a) boundary and (b) co-boundary relations, expressed as ghost simplices, for faces of 3-simplex in Fig. 3(a). Nodes in red correspond to ghost simplex (3,0,1,3) and ghost simplices in its star.

The storage cost $\mathcal{C}_t$ of graph $\mathcal{H}_t$ is the same as the storage cost of the incidence graph, restricted to one top t-simplex. We assume to encode a reference as an integer value, thus $\mathcal{C}_t = 2\sum_{p=2}^{t}(p+1)\binom{t+1}{p+1}$. The storage cost $\mathcal{C}^d$

for all graphs $\mathcal{H}_t$, is $\mathcal{C}^d = \sum_{t=1}^{d} \mathcal{C}_t$. It does not depend on the number of top simplices in Σ, and is the same for all simplicial d-complexes.

Hence, a topological query on any simplex σ is performed in two steps in any adjacency-based data structure, e.g., the IA* data structure [11]. First, we retrieve every top simplex σ' in the star of σ, and this operation is performed efficiently [11]. Then, it is possible to answer to topological queries, restricted to faces of each top simplex σ' in the star of σ, by performing breadth-first traversals of graph $\mathcal{H}_t$ (restricted to σ') in the same way as in the incidence graph [24]. For instance, in Fig. 4(b), all faces of a 3-simplex in the star of ghost simplex $(3, 0, 1, 3)$ are retrieved by a breadth-first visit of graph $\mathcal{H}_c$, and correspond to ghost simplices $(3, 0, 2, 1)$, $(3, 0, 2, 2)$, and $(3, 0, 3, 0)$ (in red). Note that nodes containing only 0 correspond to 2-faces of a top 3-simplex, which is a 3-face of itself, thus it is described by $(3, 0, 3, 0)$.

7 Experimental Results

In this section, we analyze the most interesting properties of our Mangrove TDS Library [38], available in public domain, which contains implementations of data structures in [10], including the IS and IA* data structures.

7.1 Comparisons with Other Tools

In Table 1 we compare our Mangrove TDS Library with the *OpenMesh (OM) Library* [42], the *OpenVolumeMesh (OVM) Library* [43], and the *Computational Geometry Algorithms Library (CGAL)* [14].

Table 1. The main properties of the *OpenMesh (OM) Library* [42], the *OpenVolumeMesh (OVM) Library* [43], the *Computational Geometry Algorithms Library (CGAL)* [14], and of our Mangrove TDS Library.

	OM	OVM	CGAL	Mangrove TDS
Types of shapes	cell	cell	cell	simplicial
Dimension of shapes	2	up to 3	any	any
Representation	OM [6]	OVM [32]	many	any
Extensibility	no	no	modules	yes
Non-manifolds	partially	yes	yes	yes

The *OpenMesh (OM) Library* [42] is based on the *OM* data structure [6], a variant of the HE data structure [39]. It represents only cell 2-complexes, and it cannot handle non-manifold 1-cells. Experimental results in [10, 18] show that it is about twice more expensive than the incidence graph.

The *OpenVolumeMesh (OVM) Library* [43] is based on the *OVM* data structure [32], which is almost the same as the incidence graph with orientations assigned to the 1- and 2-cells of any complex of dimension up to 3 [10].

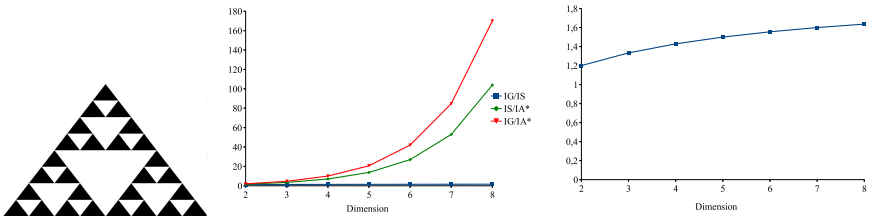
The *Computational Geometry Algorithms Library (CGAL)* [14] offers several representations of complexes, which are not designed under a common interface, like in our library. In particular, we consider the *Triangulation* [31] and the *LinearCellComplex* data structures. Note that the former is equivalent to the HE data structure, while the latter is a dimension-independent variant of the *Combinatorial Maps* [36], necessarily embedded in the Euclidean space. Experimental results in [10, 32] show that these representations, including the X-Maps [13], are more expensive than the incidence graph. The X-Maps are equivalent to the incidence graph for 2-complexes, and are about 1.2 times more verbose than the incidence graph for 3-complexes [10]. The LinearCellComplex data structure, specialized for 3-complexes, is about twice more verbose than the OVM data structure [32].

These tools are based on topological data structures, which are more verbose than the incidence graph. This latter represents complexes with a non-manifold domain. If restricted to simplicial complexes, it may be verbose and requires a large overhead, when representing manifolds [18]. In particular, it does not expose non-manifold singularities explicitly, and does not allow for their fast recognition. In fact, it provides no information regarding the link of a simplex [19], which must be reconstructed on-the-fly. The time complexity of this operation is linear in the number of simplices in the star of a simplex [10].

On the contrary, our Mangrove TDS Library, thanks to its plugin-oriented architecture, supports any topological data structure, including those representations, which are efficient in space and in recognizing non-manifold simplices, like the IS and the IA* data structures [10]. Our data structures are dimension-independent, scale well to manifolds, and are more compact than the incidence-graph [11, 19], and thus than the internal representations of other libraries. Specifically, our restrictions to simplicial 2-complexes are, respectively, 2.6 and 3.6 times more compact than the OM data structure. If restricted to simplicial 3-complexes, our representations are, respectively, 1.4 and 3.2 times more compact than the OVM data structure, thus 2.8 and 6.4 times more compact than the LinearCellComplex data structure, respectively.

Experimental results in [10] show that the IS and the IA* data structures are efficient in recognizing non-manifold simplices in the average case. In the IS data structure, any p-simplex σ is non-manifold if partial co-boundary relation $\mathcal{R}^*_{p,p+1}(\sigma)$ contains more than one $(p+1)$-simplices, one for each connected component in the link of σ. For instance, the link of any non-manifold vertex in Fig. 5(a) is formed by two connected components, i.e., two triangles in its star. In the IA* data structure, a vertex v is non-manifold if partial relation $\mathcal{R}^*_{0,p}(v)$ contains more than one top p-simplices, one for each p-cluster in the star of v, or $\mathcal{R}^*_{0,q}(v) \neq \emptyset$, for any $p \neq q$. In the same way, any $(p-1)$-simplex τ is non-manifold if $\mathcal{R}^*_{p,p-1}(\tau) \neq \emptyset$. The star of a non-manifold vertex in Fig. 5(a) is formed by two 2-clusters, i.e., two triangles in its star. The time complexities of these operations is $\mathcal{O}(1)$. In the remaining cases, it is necessary to reconstruct the topology of the link in the same way as the incidence graph.

Our Mangrove TDS Library supports complexes of any dimension, represented by the IS and the IA* data structures. We have exploited a 3Ghz Intel i7 processor with 16 Gb RAM for analyzing high dimensional versions of the *Sierpinski* shape (see Fig. 5(a) for its 2D version). Fig. 5(b) shows the ratios among their storage costs, expressed in terms of the number of references, with respect to the dimension of the shape. In our experiments, the IG and the IS data structures can be built up to dimension 8, while the IA* data structure can be built up to dimension 20. For the sake of clarity, the ratio between the storage costs of the incidence graph and the IS data structure is shown separately in Fig. 5(c). These latter tend to be very expensive and they should not be exploited in high dimensions. On the contrary, the IA* data structure is much more compact for any dimension, e.g., it is, respectively, 103 and 170 times more compact than the IG and the IS data structures for 8-complexes.



(a) (b) (c)

Fig. 5. (a) The 2D version of the *Sierpinski* shape. (b) Ratios of storage costs for the incidence graph, IS, and IA* data structures (expressed in terms of the number of references) when representing the Sierpinski shape of dimension up to 8. For the sake of clarity, (c) the behavior of the IG and IS data structures is shown separately.

7.2 Validation of Ghost Simplices

We evaluate what advantages are achieved by using ghost simplices with the IA* data structure within our framework. Recall that this mechanism can be applied to any adjacency-based representation [10], like those in [17, 20].

Table 2 shows the storage costs of the IA*, the IG, and the IS data structures, and the storage cost $\mathcal{C}^d$ of graphs $\mathcal{H}_t$ (necessary for describing ghost simplices), expressed in terms of the number of references, for the Sierpinski shapes of dimension up to 8. In our tests, $\mathcal{C}^d$ is less than 1% of the storage cost for the IA* data structure, thus it remains almost unchanged. Recall that $\mathcal{C}^d$ depends only on the dimension d of the complex, thus it is the same for any simplicial d-complex. This is interesting for complexes of dimension greater than 3, since ghost simplices are always described by four values, instead of a variable list of references, e.g., in the cell-tuples [8]. Any cell tuple is more compact than one ghost simplex for 2- and 3-complexes, but the corresponding Combinatorial Map results in a verbose representation [10].

Table 2. Comparisons among the storage cost $\mathcal{C}^d$ of auxiliary graphs $\mathcal{H}_t$, and the storage costs of the IA* ($\mathcal{S}_{IA^*}$), IS ($\mathcal{S}_{IS}$), and IG ($\mathcal{S}_{IG}$) data structures, representing Sierpinski shapes of different dimensions d with s_0 vertices and s_d maximal simplices.

d	$\mathcal{C}^d$	s_0	s_d	$\mathcal{S}_{IA^*}$	$\mathcal{S}_{IS}$	$\mathcal{S}_{IG}$
2	18	$2.8M$	$5.6M$	$22.4M$	$38M$	$44.8M$
3	74	$1.4M$	$4.2M$	$19.6M$	$68.6M$	$92.1M$
4	224	$0.7M$	$2.7M$	$14.9M$	$104.3M$	$149M$
5	596	$0.28M$	$1.4M$	$8.96M$	$123.6M$	$188M$
6	$1.5k$	$0.12M$	$0.72M$	$5.3M$	$143.1M$	$222.6M$
7	$3.5k$	$75K$	$0.52M$	$4.3M$	$228M$	$365.5M$
8	$8.1k$	$34K$	$0.27M$	$2.5M$	$260M$	$425M$

Meanwhile, thanks to auxiliary graphs $\mathcal{H}_t$, the expressive power of the IA* data structure becomes almost equivalent to the expressive power of the incidence graph and the IS data structure, including for 2- and 3-complexes. In [10] we have compared extensively the efficiency of algorithms, which answer topological queries, for all data structures we have implemented [38]. Our tests show that the IA* data structure is more efficient than other representations, even if it does not encode all simplices. Fig. 6 shows mean running times (in milliseconds) of the *BOUNDARY*, *STAR*, and *LINK* queries on the IS and the IA* data structures for all 3-complexes in our public archive [26].

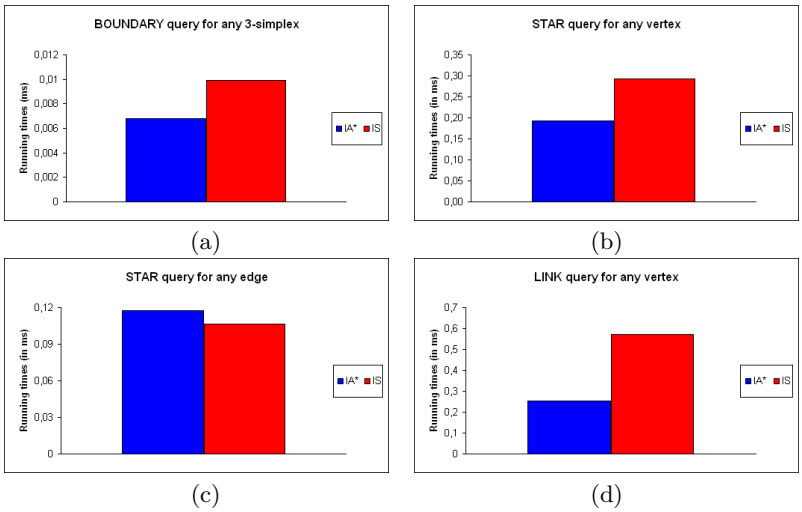


Fig. 6. Mean running times of (a) the *BOUNDARY* query on any 3-simplex, the *STAR* query on (b) vertices and on (c) edges, and (d) the *LINK* query on a vertex, if performed on the IS and the IA* data structures, representing 3-complexes in [26].

Specifically, the *BOUNDARY* query, executed on any 3-simplex, is about 30% faster in the IA* data structure. The *STAR* and the *LINK* queries, executed on any vertex, are, respectively, about 30% and 2.5 times faster in the IA* data structure. The *STAR* query, executed on any edge e, is only 10% faster in the IS data structure, despite the need to retrieve all top simplices in the star of e in the IA* data structure. The *BOUNDARY*, *STAR*, and *LINK* queries play a key role in our modeling framework. In fact, the *ADJACENCY* query is performed by combining the *BOUNDARY* and *STAR* queries. The *IS_MANIFOLD* query is based on the analysis of the link of any simplex, and thus on the *LINK* query.

8 Concluding Remarks

We have introduced the *Mangrove Topological Data Structure (Mangrove TDS)* framework, which is a tool for developing implementations of data structures for simplicial complexes of any dimension under a common programming interface. It is based on a graph-based representation of connectivity relations, that we call the *mangrove*, which can simulate the content of any topological data structure. We have also provided implicit representations, which we call the *ghost simplices*, of those topological entities, which are not directly encoded in a specific topological data structure. In order to show the validity of our contribution, we have implemented and compared six data structures [10], including the IS [19] and the IA* [11] data structures on simplicial complexes. Our implementations are contained in the *Mangrove TDS Library* [38], released in public domain. Our tests show that the majority of the queries are more efficient when using the IA* data structure, even if this latter does not encode all topological entities. This is due to the ghost simplices, which improve its expressive power (and of any topological data structure with similar properties) without introducing a relevant overhead. This is important, since the IA* data structure is one of the most compact representations for non-manifold shapes in the literature [10]. Moreover, these representations are very efficient for high-dimensional complexes, since they are always formed by the same number of references, despite the dimension of the complex.

We have demonstrated so far the IA* data structure and other representations for simplicial complexes. Our current implementation (which is not ready to be released), already supports cell complexes in which the number of faces of a cell is a constant when fixing the dimension of the complex, like unstructured quad and hex meshes. We are extending and testing our data structures with these complexes.

On the other side, our library does not support editing operators, e.g., stellar operators, edge or vertex-pair collapse on simplicial complexes, and Euler operators for cell complexes. We are currently working in this direction. In particular, we are designing and implementing a new set of Euler operators on cell complexes [15], which preserve the simplicial homology, like Betti numbers

and generators. These operations are useful to improve the efficiency of homology computations. We are also developing multi-resolution representations based on homology-preserving and homology-modifying operators.

Finally, our Mangrove TDS framework may be extended in order to represent shapes, which are not necessarily discretized by simplicial complexes, including for applications in high dimensions [3, 5].

Acknowledgments
The *Sierpinski* shape is courtesy of Professor Vijay Natarajan, Indian Institute of Science, Bangalore, India. Authors want to thank anonymous reviewers for their useful comments. This work has been supported by the Italian Ministry of Education and Research under the PRIN 2009 program, and by the National Science Foundation under grant number IIS-1116747.

References

1. T. J. Alumbaugh and X. Jiao. Compact Array-based Mesh Data Structures. In *Proc. of the 14th Int. Mesh. Rnd.*, pages 485–503, 2005.
2. V. Arvind, D. Bireswar, and J. Kobler. The Space Complexity of k-Tree Isomorphism. In *Proc. of the 18th Int. Conf. on Alg. and Comp.*, 2007.
3. D. Attali, A. Lieutier, and D. Salinas. Efficient Data Structure for Simplifying Simplicial Complexes in High Dimensions. *Int. Journal of Comp. Geom. and Appl.*, 22(4):279–303, 2012.
4. D. Blandford, G. Blelloch, D. Cardoze, and C. Kadow. Compact Representations of Simplicial Meshes in Two and Three Dimensions. In *Proc. of the 12th Int. Mesh. Rnd.*, pages 135–146, 2003.
5. J.-D. Boissonnant and C. Maria. The Simplex Tree: an Efficient Data Structure for General Simplicial Complexes. In *Proc. of the 20th Europ. Symp. on Alg.*, volume 7501 of *Lecture Notes in Computer Science*, pages 731–742, 2012.
6. M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt. Open-Mesh: a Generic and Efficient Polygon Mesh Data Structure. In *Proc. of the OpenSG Symp.*, 2002.
7. M. Brewer, L. Freitag, P. M. Knupp, T. Leurent, and D. Melander. The Mesquite Mesh Quality Improvement Toolkit. In *Proc. of the 12th Int. Mesh. Rnd.*, pages 239–250, 2003.
8. E. Brisson. Representing Geometric Structures in d-dimensions: Topology and Order. In *Proc. of the Symp. on Comp. Geom.*, pages 218–227, 1989.
9. S. Campagna, L. Kobbelt, and H.-P. Seidel. Directed Edges: a Scalable Representation for Triangle Meshes. *Jour. of Graph. Tools*, 3(4), 1998.
10. D. Canino. *Tools for Modeling and Analysis of Non-Manifold Shapes.* PhD thesis, Department of Computer Science, University of Genova, Italy, 2012.
11. D. Canino, L. De Floriani, and K. Weiss. IA*: an Adjacency-based Representation for Non-Manifold Simplicial Shapes in Arbitrary Dimensions. *Comp. & Graph.*, 35(3):747–753, 2011.
12. D. Cardoze, G. Miller, and T. Phillips. Representing Topological Data Structures using Cell-Chains. In *Proc. of the 4th Int. Conf. on Geom. Mod. and Proc.*, pages 248–266, 2006.

13. D. Cazier and P. Kraemer. X-Maps: an Efficient Model for Non-manifold Modeling. In *Proc. of the Shape Mod. Int. Conf.*, pages 226–230, 2010.
14. *Computational Geometry Algorithms Library*, 2011. `http://www.cgal.org`.
15. L. Comic, L. De Floriani, and F. Iuricich. Multi-resolution Cell Complexes based on Homology-Preserving Euler Operators. In *Discr. Geom. for Comp. Imag.*, volume 7749 of *Lect. Notes in Comp. Sci.*, pages 323–334. Springer, 2013.
16. L. De Floriani, D. Greenfieldboyce, and A. Hui. A Data Structure for Non-manifold Simplicial d-complexes. In *Proc. of the Symp. on Geom. Proc.*, pages 83–92, 2004.
17. L. De Floriani and A. Hui. A Scalable Data Structure for Three-dimensional Non-manifold Objects. In *Proc. of the Symp. on Geom. Proc.*, pages 72–82, 2003.
18. L. De Floriani and A. Hui. Data Structures for Simplicial Complexes: an Analysis and a Comparison. In *Proc. of the Symp. on Geom. Proc.*, pages 119–128, 2005.
19. L. De Floriani, A. Hui, D. Panozzo, and D. Canino. A Dimension-independent Data Structure for Simplicial Complexes. In *Proc. of the 19th Int. Mesh. Rnd.*, pages 403–420, 2010.
20. L. De Floriani, P. Magillo, E. Puppo, and D. Sobrero. A Multi-resolution Topological Representation for Non-manifold Meshes. *CAD Jour.*, 36(2):141–159, 2004.
21. A. Di Carlo, F. Milicchio, A. Paoluzzi, and V. Shapiro. Solid and Physical Modeling with Chain Complexes. In *Proc. of the Symp. on Sol. and Phys. Mod.*, pages 73–84, 2007.
22. D. Dobkin and M. Laszlo. Primitives for the Manipulation of Three-dimensional Subdivision. *Algor.*, 5(4):3–32, 1989.
23. M. S. Ebeida and P. M. Knupp. LBMD: a Layer-based Mesh Data Structure tailored for Generic Implementations. In *Proc. of the 20th AIAA Comp. Fluid Dyn. Conf.*, 2011.
24. H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer, 1987.
25. R. V. Garimella. Mesh Data Structure Selection for Mesh Generation and FEA Applications. *Int. Jour. of Num. Meth. in Eng.*, 55(4):451–478, 2002.
26. *Non-Manifold Meshes Repository*, 2009. Department of Computer Science, Genova, Italy, `http:://indy.disi.unige.it/nmcollection`.
27. T. Gurung, D. Laney, P. Lindstrom, and J. Rossignac. SQuad: a Compact Representation for Triangle Meshes. *Comp. Graph. For.*, 30(2):355–364, 2011.
28. T. Gurung, M. Luffel, P. Lindstrom, and J. Rossignac. LR: Compact Connectivity Representation for Triangle Meshes. *ACM Trans. on Graph.*, 30(4), 2011.
29. T. Gurung, M. Luffel, P. Lindstrom, and J. Rossignac. Zipper: a Compact Connectivity Data Structure for Triangle Meshes. *Comp.-Aid. Des.*, 45(2):262–269, 2013.
30. T. Gurung and J. Rossignac. SOT: Compact Representation for Tetrahedral Meshes. In *Proc. of the Symp. on Sol. and Phys. Mod*, pages 79–88, 2009.
31. L. Kettner. Using Generic Programming for Designing a Data Structure for Polyhedral Surfaces. *Comp. Geom. - Theory and Appl.*, 1(13):65–90, 1999.
32. M. Kremer, D. Bommes, and L. Kobbelt. OpenVolumeMesh: a Versatile Index-based Data Structure for 3D Polytopal Complexes. In *Proc. of the 21st Int. Mesh. Rnd.*, pages 531–548, 2012.
33. M. Lage, T. Lewiner, H. Lopes, and L. Velho. CHF: a Scalable Topological Data Structure for Tetrahedral Meshes. In *Proc. of the 18th Braz. Symp. on Comp. Grap. and Img. Proc.*, pages 349–356, 2005.
34. F. Ledoux, J.-C. Weill, and Y. Bertrand. GDMS: a Generic Mesh Data Structure. In *Proc. of the 17th Int. Mesh. Rnd.*, 2008.

35. S. Lee and K. Lee. The Partial-Entity Structure: a Fast and Compact Non-manifold Boundary Representation based on Partial Topological Entities. In *Proc. of the Symp. on Sol. and Phys. Mod.*, pages 159–170, 2001.
36. P. Lienhardt. N-dimensional Generalized Combinatorial Maps and Cellular Quasi-manifolds. *Int. Jour. of Comp. Geom. and Appl.*, 4(3):275–324, 1994.
37. H. Lopes and G. Tavares. Structural Operators for Modeling 3-manifolds. In *Proc. of the Symp. on Sol. Mod. and Appl.*, pages 10–18, 1997.
38. *Mangrove Topological Data Structure Library*, 2012. `http://mangrovetds.sourceforge.net`.
39. M. Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, 1987.
40. S. Mc-Mains. *Geometric Algorithms and Data Representation for Solid Free-form Fabrication*. PhD thesis, University of California, Berkeley, USA, 2000.
41. A. Nabutovsky. Geometry of the Space of Triangulations of a Compact Manifold. *Comm. in Math. Phys.*, 181:303–330, 1996.
42. *OpenMesh Library*, 2002. `http://openmesh.org`.
43. *OpenVolumeMesh Library*, 2012. `http://openvolumemesh.org`.
44. A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-Independent Modeling with Simplicial Complexes. *ACM Trans. on Graph.*, 12(1):56–102, 1993.
45. J.-F. Remacle and M. S. Shephard. An Algorithm Oriented Mesh Database. *Int. Jour. of Comp. Geom. and Appl.*, 58(2):393–374, 2003.
46. J. Rossignac, A. Safonova, and A. Szymczak. 3D Compression made Simple: Edge-breaker on a Corner Table. In *Proc. of the Shape Mod. Int. Conf.*, pages 278–283, 2001.
47. E. S. Seol and M. S. Shephard. Efficient Distributed Mesh Data Structure for Parallel Automated Adaptative Analysis. *Eng. with Comp.*, 22(3):197–213, 2006.
48. D. Sieger and M. Botsch. Design, Implementation, and Evaluation of the Surface_Mesh Data Structure. In *Proc. of the 20th Int. Mesh. Rnd.*, pages 533–550, 2011.
49. T. Tautges. MOAB-SD: Integrated Structured and Unstructured Mesh Representation. *Eng. with Comp.*, 20(3), 2004.
50. *Visual Computing Graphics Library*, 2004. `http://vcg.isti.cnr.it/`.
51. K. Weiler. The Radial-Edge data structure: a Topological Representation for Non-manifold Geometric Boundary Modeling. In *Geom. Mod. for CAD Appl.*, pages 3–36. Elsevier, 1998.
52. P. K. Wright. *21st Century Manifacturing*. Prentice Hall, 2001.
53. L. Zeng, Y.-J. Liu, S. H. Lee, and M. Yuen. Q-Complex: Efficient Non-Manifold Boundary Representation with Inclusion Topology. *Comp.-Aid. Des.*, 44(11):1115–1126, 2012.