A Dimension-Independent Data Structure for Simplicial Complexes

Leila De Floriani[1], Annie Hui[2], Daniele Panozzo[1], and David Canino[1]

[1] Department of Computer Science, University of Genova, Italy
deflo@disi.unige.it, panozzo@disi.unige.it, canino@disi.unige.it
[2] Department of Computer Science, University of Maryland, MD, USA
hui@cs.umd.edu

Summary. We consider here the problem of representing non-manifold shapes discretized as d-dimensional simplicial Euclidean complexes. To this aim, we propose a dimension-independent data structure for simplicial complexes, called the *Incidence Simplicial (IS)* data structure, which is scalable to manifold complexes, and supports efficient navigation and topological modifications. The IS data structure has the same expressive power and exibits performances in query and update operations as the *incidence graph*, a widely-used representation for general cell complexes, but it is much more compact. Here, we describe the IS data structure and we evaluate its storage cost. Moreover, we present efficient algorithms for navigating and for generating a simplicial complex described as an IS data structure. We compare the IS data structure with the incidence graph and with dimension-specific representations for simplicial complexes.

1 Introduction

Simplicial complexes are commonly used to represent objects in many applications, including finite element analysis, solid modeling, animation, terrain modeling and visualization of scalar and vector fields [7, 8, 15, 19, 20]. Informally, a *manifold* (with boundary) $\mathcal{M}$ is a compact and connected subset of the Euclidean space such that the neighborhood of each point of $\mathcal{M}$ is homeomorphic to an open ball, or to an open half-ball. Objects, that do not fulfill this property at one or more points, are called *non-manifold*, while they are called *non-regular* if they also contain parts of different dimensions.

Existing modeling tools are generally designed to handle only shapes with a manifold domain, since they are topologically simpler. However, non-manifold and non-regular objects arise in several applications. For instance, they are produced through the idealization process when preparing an object for finite element analysis. To guarantee acceptable computation times, not only details having a low impact on the object behavior are removed, but also subparts of the object are idealized. For instance, parts presenting a beam behavior may be substituted with 1-dimensional entities, and parts presenting a plate behavior may be replaced by 2-dimensional surfaces. Thus, the resulting object will contain non-manifold singularities and parts of different dimensions.

In the literature, both dimension-specific and dimension-independent data structures have been developed for cell and simplicial complexes: many of them have domains that are limited to manifold shapes [7]. A widely used data structure is the *Incidence Graph (IG)* [10], developed for cell complexes in arbitrary dimensions. However, if restricted to simplicial complexes, the incidence graph results in a verbose representation, which also does not scale well to the manifold case, i.e., it has a large overhead when representing a manifold shape. Scalability is an important property for data structures for cell and simplicial complexes, since non-manifold objects often present few non-manifold singularities.

Here, we propose a dimension-independent data structure for simplicial complexes, that we call the *Incidence Simplicial* (IS) data structure. The IS data structure encodes all the simplices of a simplicial complex and a subset of the relations encoded by the incidence graph. We show that this information is sufficient to retrieve all the relations among the elements of a simplicial d-complex efficiently. The algorithms for retrieving such relations are the basic traversal tools for navigating in a complex. The ability to navigate is essential to any modeling and update operations, as such operations require the knowledge of the local neighborhood of every simplex. We show that the IS data structure scales well to the manifold case. It has the same representation power and comparable performances in querying operations as the incidence graph, but it is much more compact. It represents all the simplices explicitly and uniquely, as required in applications where attributes must be attached to all the simplices.

Thus, novel contributions of this paper are:

- the *Incidence Simplicial* data structure, a dimension-independent and scalable data structure for encoding d-dimensional simplicial complexes;
- efficient navigation algorithms for retrieving topological relations from the IS data structure;
- an algorithm for generating an IS data structure from a non-topological representation of a d-dimensional simplicial complex.

The remainder of this paper is organized as follows. In Sect. 2, we summarize some background notions on simplicial complexes and on topological relations. In Sect. 3, we review related work. In Sect. 4, we describe the IS data structure, while in Sect. 5 we discuss the implementation of the IS data structure and its storage cost. In Sect. 6, we describe an algorithm for generating an IS data structure from an unstructured collection of simplices, while in Sect. 7 we present navigation algorithms for efficiently retrieving topological relations among the simplices of a complex. In Sect. 8, we compare the IS data structure with the incidence graph and with dimension-specific data structures for simplicial complexes. Finally, in Sect. 9, we draw some concluding remarks and discuss future work.

2 Background Notions

2.1 Simplicial Complexes

A Euclidean simplex σ of dimension d is the convex hull of $d+1$ linearly independent points in the n-dimensional Euclidean space E^n, with $d \leq n$. We simply call a *Euclidean d-simplex* a *d-simplex*: a 0-simplex is a *vertex*, a 1-simplex an *edge*, a 2-simplex a *triangle*, and a 3-simplex a *tetrahedron. d* is called the *dimension* of σ and is denoted $dim(\sigma)$. Any Euclidean k-simplex σ', with $k < d$, generated by a set $V_{\sigma'} \subseteq V_\sigma$ of cardinality $k+1 \leq d$, is called a *k-face* of σ. Where no ambiguity arises, the dimension of σ' can be omitted and σ' is simply called a *face* of σ.

A finite collection Σ of Euclidean simplices forms a *Euclidean simplicial complex* when (i) for each simplex $\sigma \in \Sigma$, all the faces of σ belong to Σ, and (ii) for each pair of simplices σ and σ', either $\sigma \cap \sigma' = \emptyset$ or $\sigma \cap \sigma'$ is a face of both σ and σ'. The maximal dimension d of the simplices in Σ is called the *order*, or the *dimension* of complex Σ. The *domain*, or *carrier*, of a d-dimensional Euclidean simplicial complex (also called a simplicial d-complex) Σ embedded in E^n, with $d \leq n$, is the subset of E^n defined by the union, as point sets, of all the simplices in Σ.

The *(combinatorial) boundary* of a simplex σ is the set of all the faces of σ in Σ, different from σ itself. The *star* of a simplex σ is the set of simplices in Σ that have σ as a face. Any simplex σ such that the star of σ contains only σ is called a *top* simplex. The *link* of a simplex σ is the set of all the faces of the simplices in the star of σ which are not incident in σ.

Two simplices are *incident* to each other if one of them is a face of the other, while they are *k-adjacent* if they share a k-face: in particular, two vertices are called *adjacent* if they are both incident at a common edge. An *h-path* is a sequence of $(h+1)$-simplices $(\sigma_i)_{i=0}^k$ in a simplicial complex Σ such that two consecutive simplices σ_{i-1} and σ_i in the sequence are h-adjacent. Two simplices σ and σ^* are *h-connected* when there exists an h-path $(\sigma_i)_{i=0}^k$ such that σ is a face of σ_0 and σ^* is a face of σ_k. A subset Σ^* of a complex Σ is called *h-connected* if and only if any two simplices of Σ^* are h-connected. Any maximal h-connected sub-complex of a complex Σ is called an *h-connected component* of Σ.

A d-complex Σ in which all top simplices are d-simplices is called *regular.* A regular $(d-1)$-connected d-complex in which the star of all $(d-1)$-simplices consists of one or two simplices is called a *(combinatorial) pseudo-manifold.* Note that a carrier of a pseudo-manifold is not necessarily a manifold object. We will call a *manifold simplicial complex* a pseudo-manifold such that the link of any 0-simplex, with $k < d-1$, is homeomorphic to the $(d-1)$-sphere.

In a simplicial d-complex Σ, a *p-chain* c (with $0 \leq p \leq d$) is a linear combination of $p+1$ simplices σ_i, defined as $c = \sum_{i=0}^{p} a_i \sigma_i$, where $a_i \in Z$. A 0-simplex can be expressed as a 0-chain, while a p-simplex σ (with $p > 0$) can be expressed as a p-chain, generated by its faces λ_i, with $i = 0, \ldots, p$. We can define the *boundary* of a p-simplex σ as $\delta_p(\sigma) = \sum_{i=0}^{p} a_i \delta_{p-1}(\lambda_i)$ by recursively computing the boundary of each λ_i, with $i = 0, \ldots, p$ since λ_i is a $(p-1)$-chain by definition. The coefficients a_i of a p-chain do not have a geometric interpretation, except when $a_i = \pm 1$. In this case, we can relate them to the orientation of the simplices: a formal definition of this concept is available in [1]. If $a_i = 1$, then we can consider the positive orientation of a simplex σ_i, otherwise we can consider the opposite one. Hence, we can define the *oriented boundary* of a p-simplex σ, generated by its faces λ_i, as $\delta_p(\sigma) = \sum_{i=0}^{p} (-1)^i \delta_{p-1}(\lambda_i)$.

2.2 Topological Relations

The connectivity information among the entities in a cell or in a simplicial complex is expressed through *topological relations*. Data structures for cell and simplicial complexes can be described formally in terms of the topological entities and relations they encode. We define topological relations for the case of a cell complex (since a simplicial complex can be seen as a special case of a cell complex). Informally, a cell complex is a collection of cells, homeomorphic to a closed or an open k-ball, such that the boundary of each cell is the union of lower-dimensional cells. We consider a cell d-complex Γ and a p-cell $\gamma \in \Gamma$, with $0 \leq p \leq d$. We can define *topological relations* as follows:

- *Boundary relation* $R_{p,q}(\gamma)$, with $0 \leq q \leq p-1$, consists of the set of q-cells which are faces of γ.
- *Coboundary relation* $R_{p,q}(\gamma)$, with $p+1 \leq q \leq d$, consists of the set of q-cells incident in γ.
- *Adjacency relation* $R_{p,p}(\gamma)$, with $p > 0$, consists of the set of p-cells in Γ that are $(p-1)$-adjacent to γ.
- *Adjacency relation* $R_{0,0}(\gamma)$, where γ is a vertex, consists of the set of vertices that are adjacent to γ through a 1-cell (an edge).

We call *constant* any relation which involves a constant number of entities, while relations which involve a variable number of entities are called *variable*. In general, coboundary and adjacency relations are variable, while boundary relations are constant in simplicial complexes. We consider an algorithm for retrieving a topological relation R to be *optimal* if it retrieves R in time linear in the number of the involved entities. If the retrieval of a relation requires examining the star of all the cells adjacent to or on the boundary of the query cell, then we say that the data structure offers a *local* support for the retrieval of that relation. If the retrieval of a relation requires examining all the cells of a specific dimension, then the data structure does not support an efficient retrieval of that relation.

3 Related Work

Dimension-independent data structures have been proposed for encoding d-dimensional manifold cell complexes. Examples are the *cell-tuple* [2], and the *n-G-map* [16]. The *incidence graph* [10] represents a cell complex by encoding all the cells of the complex and a subset of their boundary and coboundary relations. It provides a complete and verbose description of the complex. The *simplified incidence graph* [5] is a simplified version of the incidence graph for simplicial complexes, but it is not suitable for supporting efficient updates of the complex. The *indexed data structure with adjacencies* [20] is a dimension-independent data structure for pseudo-manifold simplicial d-complexes embedded in the d-dimensional Euclidean space. It encodes only the d-simplices together with boundary relation $R_{d,0}$ and adjacency relation $R_{d,d}$.

Several dimension-specific data structures have been developed for manifold 2-dimensional cell complexes [7], whereas some of them are specific for triangle meshes, e.g. the *Corner table* [23] and the *star vertex* [13] data structures. Data structures for non-manifold, non-regular, 2-dimensional cell complexes have been proposed for modeling non-manifold solids. The *partial entity* data structure [15] is the most scalable to the manifold case, but it is still verbose when applied to simplicial 2-complexes [7]. Dimension-specific data structures have been also proposed for encoding simplicial 2-complexes [3, 5, 19]. The *loop edge-use* [19] and the *directed edge* [3] data structures are for regular simplicial complexes in which non-manifold singularities occur only at edges. The former is a specialization of the partial entity data structure, and the latter is an extension of the *half-edge* data structure [18] to non-manifold shapes. The *Triangle-Segment (TS)* data structure [8] is a compact data structure for non-manifold simplicial 2-complexes embedded in 3-dimensional Euclidean space, encoding only the top simplices and the vertices of the complex. A comparison among such data structures, presented in [7], shows that the TS data structure requires about half of the space required by the loop edge-use and by the directed edge data structures.

To the extend of our knowledge, very few representations have been proposed in the literature for 3-dimensional manifold complexes, i.e., the *facet-edge* [9] and the *handle-face* [17] data structures. Both of them describe 3-dimensional cells implicitly by encoding the manifold complexes that form their boundary. The *handle-face* structure is an extension of the *handle-edge* data structure [4] for 2D complexes. In [14], a scalable data structure for manifold tetrahedral complexes has been proposed, which extends the *Corner table* [23] to the 3D case. An efficient extension of the *Corner table* to tetrahedral meshes has been recently proposed in [11]. In [6], we have proposed a compact and scalable data structure for arbitrary 3-dimensional simplicial complexes embedded in the 3-dimensional Euclidean space, called the *Non-Manifold Indexed data structure with Adjacencies (NMIA)*, which extends the indexed data structure with adjacencies to arbitrary complexes.

An alternative approach to the design of non-manifold data structures consists of decomposing a non-manifold object into simpler and more manageable parts [7]. Such techniques deal with the decomposition of the boundary of a regular object into two-manifold parts, and all of them are for cell complexes. The representation in [21] deals with non-regular objects as well. In [12], we have proposed a data structure for simplicial 3-complexes, called the *Double Level Decomposition* (DLD) data structure, based on a decomposition for 3-dimensional non-manifold and non-regular objects into nearly manifold components. The DLD is a two-level representation in which the higher level encodes the decomposition, and the lower level encodes each component as an indexed data structure with adjacencies. We have shown that the DLD has similar performances as the NMIA data structure. Unfortunately, it is more complex to update, since the decomposition of the complex needs to be reconstructed at each update.

4 The Incidence Simplicial data structure

In this section, we introduce a new dimension-independent data structure for representing Euclidean simplicial complexes in arbitrary dimensions, that we call the *Incidence Simplicial* (IS) data structure. The IS data structure encodes all the simplices of a d-dimensional simplicial complex Σ embedded in the n-dimensional Euclidean space (with $d \leq n$) and the following relations:

- the boundary relation $R_{p,p-1}(\sigma)$ for each p-simplex σ, where $0 < p \leq d$;
- the partial coboundary relation $R_{p,p+1}(\sigma)$, denoted as $R^*_{p,p+1}(\sigma)$, for each p-simplex σ, with $0 \leq p < d$: it consists of one arbitrarily selected $(p+1)$-simplex for each connected component in the link of σ.

The IS data structure stores the *orientation* of the simplices forming it. Given a p-simplex σ generated by the ordered set of vertices $V = [v_0, \ldots, v_p]$, a face λ_i of σ is expressed as $\lambda_i = [v_0, \ldots, \hat{v}_i, \ldots, v_p]$, where we discard the vertex v_i with $i = 0, \ldots, p$.

In general, the partial coboundary relation $R^*_{d-1,d}(\sigma)$ is the same as the coboundary relation $R_{d-1,d}(\sigma)$ for each $(d-1)$-simplex σ. If the domain of Σ is a manifold, then $R^*_{p,p+1}(\sigma)$ contains just one $(p+1)$-simplex since the link of σ consists of one single connected component. When $d = n$, any d-dimensional simplicial complex is a pseudo-manifold in the d-dimensional Euclidean space. Consequently, every $(d-1)$-simplex is shared by at most two d-simplices. Fig. 1 shows two examples of partial coboundary relations in the IS data structure. In Fig. 1(a), the link of vertex v consists of two connected components, as shown in Fig. 1(b). Thus, partial coboundary relation $R^*_{0,1}(v)$ consists of $\{we, e\}$, where e is an edge of triangle df. In Fig. 1(c), the link of an edge e (shown in Fig. 1(d)) is composed of three connected components corresponding to the triangle df and to the tetrahedra t_1 and t_2, incident in e. Thus, the partial coboundary relation $R^*_{1,2}(e)$ consists of just three elements, namely $\{df, f_1, f_2\}$, where f_1 is a face of t_1 and f_2 is a face of t_2.

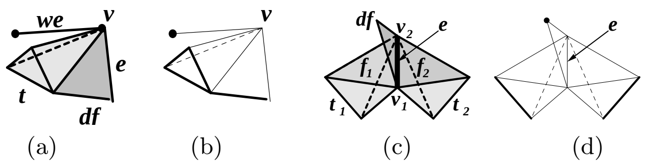


Fig. 1. (a) An example of a non-manifold vertex v, whose link is shown in (b) with thick lines. (c) An example of a non-manifold edge e, whose link is shown in (d).

5 Implementation of the IS data structure

In the IS data structure describing a simplicial d-complex Σ, all simplices of the same dimension p (with $0 \leq p \leq d$) are stored in a dynamic array, that we call *SimplexesContainer*, where each location stores a simplex. The *SimplexesContainer* array supports a garbage collector mechanism and, thus, update operations (such as simplex insertion or removal).

Each p-simplex σ has an unique identifier in the IS data structure, formed by the pair (p, i), where p is the dimension of σ and i is the position of σ in the *SimplexesContainer* where it is stored. This identifier is called *Simplex-Pointer* and allows accessing a simplex in constant time. A flag is associated with each simplex for marking it as visited during traversal algorithms. Some application-dependent attributes, like Euclidean coordinates and field values, can be associated to each simplex.

In order to encode the simplex orientation, we order the list of vertices in the simplicial complex Σ according to the lexicographic order on their identifiers. Given two vertices v_i and v_j, respectively identified by the *SimplexPointers* $(0, i)$ and $(0, j)$, then we say that $v_i < v_j$ if and only if $i < j$. Consequently, the orientation of the simplices is imposed by the input list of vertices in a simplicial complex and it must be enforced when building the data structure (see Sect. 6).

For each p-simplex σ, we store boundary relation $R_{p,p-1}(\sigma)$. We use an array of dimension $p+1$, where each element stores the index of a $(p-1)$-face in the corresponding *SimplexesContainer* array. We also encode the orientation of the simplex boundary. The index of a face λ_i of a p-simplex σ is stored as the i-th element in boundary relation $R_{p,p-1}(\sigma)$. Recall that a face λ_i is the simplex defined by all the vertices of σ with the exception of vertex v_i, and that a face λ_i precedes a face λ_j in $R_{p,p-1}(\sigma)$ if and only vertex v_i precedes vertex v_j in the lexicographic order of the vertices of Σ.

For each p-simplex σ, we also store partial coboundary relation $\mathcal{R}^*_{p,p+1}(\sigma)$, formed by one arbitrarily-selected $(p+1)$-simplex for each connected component in the link of σ. We use a variable-sized array, where each element stores the index of the coboundary simplex in the corresponding *SimplexesContainer*.

In what follows, we evaluate the storage cost of the IS data structure representing a simplicial d-complex Σ. We consider only the cost of encoding the topological information and we assume that both indices and pointers are described through an integer value. In our analysis, we denote:

- the number of p-simplices in Σ (where $0 \leq p \leq d$) as n_p;
- the total number of connected components in the link of a simplex σ (where $dim(\sigma) < d$) as $k(\sigma)$;
- the total number of connected components summed over the links of all the p-simplices in Σ as $K_p = \sum_{\sigma | dim(\sigma)=p} k(\sigma)$ with $0 \leq p \leq d$.

In order to encode all the p-simplices, n_p pointers to the records describing the simplices and a pointer for each *SimplexesContainer* are required. This results in $d + 1 + \sum_p n_p$ integer values, with $0 \leq p \leq d$. For each p-simplex (with $0 < p \leq d$), there are $p + 1$ simplices of dimension $p - 1$ in $R_{p,p-1}(\sigma)$. Thus, we need $\sum (p + 1) n_p$ integer values in order to store the boundary relations for all simplices in Σ. Moreover, for each p-simplex (with $0 \leq p < d$), there are $k(\sigma)$ simplices of dimension $p + 1$ in $R^*_{p,p+1}(\sigma)$. Thus, we need $\sum K_p$ integer values in order to store all partial coboundary relations for all simplices in Σ. In summary, the storage cost of the *IS* data structure is equal to $d + 1 + \sum_{0 \leq p \leq d} n_p + \sum_{0 < p \leq d} (p + 1) n_p + \sum_{0 \leq p < d} K_p$.

If the simplicial complex Σ is manifold, then we obtain a more compact representation for all partial coboundary relations, while the storage cost for the simplices and for all boundary relations does not change. For all the simplices σ such that $dim(\sigma) < d - 1$, then $k(\sigma) = 1$, while $k(\sigma) \leq 2$ for all the $(d - 1)$-simplices. The above formula in the case of a manifold simplex becomes $d + 1 + \sum_{0 \leq p \leq d} n_p + \sum_{0 < p \leq d} (p + 1) n_p + 2n_{d-1} + \sum_{0 \leq p \leq d-2} n_p$.

6 Building an IS data structure

The most common exchange format for a d-dimensional simplicial complex Σ consists of a collection of top simplices described by their vertices. This representation is known as a *soup of top simplices.* In this section, we describe how to generate the IS data structure from such representation.

The input format provides the list of the vertices of the simplicial complex Σ. Each top p-simplex of Σ is described by the indexes of its $(p + 1)$ vertices in the input vertex list. The orientation of the simplices in Σ is well defined by the input vertices, as described in Sect. 5, since a vertex is identified by its position in such list. Since a soup of simplices describes only the top simplices, we first need to generate all the simplices in Σ and to establish the topological relations among them. This is achieved in four steps by analyzing all the p-simplices in decreasing order of their dimension:

1. For each p-simplex σ, all the $p+1$ faces σ_i of dimension $p-1$ are generated and stored in an auxiliary data structure. Each face λ_i is defined by all the vertices of σ with the exception of the vertex in position i.

2. All the $(p-1)$-simplices generated at step 1 are sorted by lexicographic order of their vertices. In this way, duplicated simplices are removed. Each simplex σ is stored in the corresponding *SimplexesContainer* array and is given a unique identifier, i.e., its *SimplexPointer*.
3. All the $(p-1)$-faces λ_i of a p-simplex σ are considered in order to compute the boundary relation $R_{p,p-1}$ and the complete coboundary relation $R_{p-1,p}$. The identifier of σ is added to the coboundary relation $R_{p-1,p}(\lambda_i)$, while the identifier of λ_i is stored in the i-th position in the boundary relation $R_{p,p-1}(\sigma)$.
4. Partial coboundary relation $R^*_{p-1,p}(\sigma)$ is computed for each $(p–1)$-simplex σ from the corresponding complete coboundary $R_{p-1,p}(\sigma)$ relation, as detailed below.

The computation of the partial coboundary relation of a p-simplex σ in Step 4 is based on the topological information available in the intermediate structure obtained at Step 3, i.e., boundary relations $R_{p,p-1}(\sigma)$ and coboundary relations $R_{p,p+1}(\sigma)$, for each p-simplex σ.

For a p-simplex σ, with $p < d-1$, we identify the connected components of the link of σ and we encode a $(p+1)$-simplex incident in σ for each component of the link. Fig. 2(a) shows an example of the star of a vertex v, where there are two connected components in the link. We consider the graph, that we call the *star-graph* of σ, in which the nodes are the simplices in the star of σ and the arcs correspond to the boundary $R_{p,p-1}$ and coboundary $R_{p,p+1}$ relations between the simplices in the star, restricted to the nodes in the star-graph. Figs. 2(b) and 2(c) show the boundary and coboundary arcs of the star-graph associated with vertex v (see Fig. 2(a)). The two sets of arcs are shown separately, for the sake of clarity. It can be easily seen that the connected components of the link of a p-simplex σ are the same as the biconnected components of the star-graph. For instance, the star-graph of vertex v in Fig. 2 has two biconnected components. We also call the nodes in the star-graph corresponding to j-simplices nodes at level j.

In order to identify all biconnected components in the star-graph, we traverse the nodes belonging to levels $(p+1)$ and $(p+2)$ in the star-graph, by using the arcs corresponding to relations $R_{p,p+1}$, $R_{p+1,p+2}$ and $R_{p+2,p+1}$. We attach a unique label to each biconnected component. We start the traversal from an unmarked node representing a $(p+1)$-simplex τ incident in σ and we assign node τ to a new component by marking it with a new label. For each node representing a $(p+2)$-simplex θ in $R_{p+1,p+2}(\tau)$, we retrieve the nodes corresponding to the $(p+1)$-simplices in its boundary and we continue the traversal. In this way, all the nodes corresponding to the $(p+1)$-simplices μ in $R_{p+2,p+1}(\theta)$, that are incident at σ, are marked with the same label. The graph traversal is recursively repeated for all nodes corresponding to simplices in $R_{p+1,p+2}(\mu)$ until all nodes associated with the $(p+1)$-simplices incident at σ and belonging to the same biconnected component of the star-graph are visited. Then, for each biconnected component in the star-graph, one $(p+1)$-

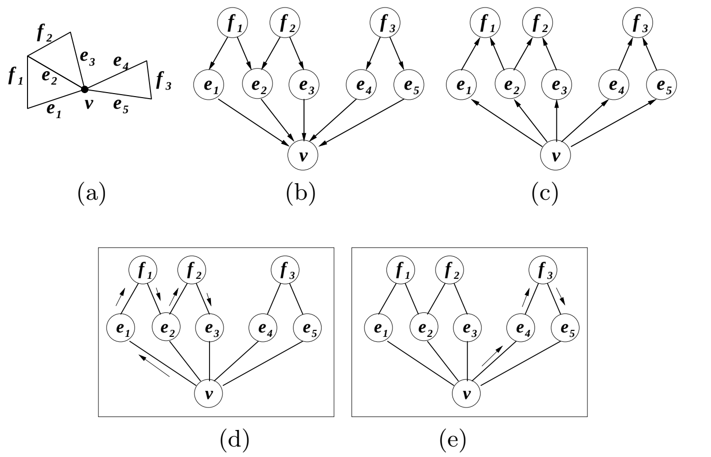


Fig. 2. The topological relations of simplices in the star of vertex v encoded as a star-graph: (a) the star of vertex v; (b) the boundary relations $R_{1,0}$ and $R_{2,1}$; (c) the coboundary relations $R_{0,1}$ and $R_{1,2}$ computed as intermediate results in the IS construction; (d) a traversal of one component in the star of v by using only the relations $R_{0,1}, R_{1,2}$ and $R_{2,1}$; (e) a traversal of the other component in the star of v.

simplex is selected as an element of $R^*_{p,p+1}(\sigma)$. Fig. 2(d) shows the traversal of one biconnected component of the star-graph of vertex v for the complex depicted in Fig. 2(a), while Fig. 2(e) shows the traversal of the other biconnected component in the same star-graph. We note that we do not need to compute the complete star-graph of σ, but only the nodes at levels $(p+1)$ and $(p+2)$ and their boundary and coboundary arcs in the star-graph.

We can evaluate the time complexity of the IS construction as follows. At step 1, at any level p, the total number of $(p-1)$-simplices is $(p+1) \cdot n_p$, where n_p is the number of p-simplices in complex Σ. Thus, the creation of the $(p-1)$-simplices is linear with respect to the number of p-simplices. The time required for sorting all the $(p-1)$-simplices at step 2 is $O(n_p log(n_p))$. The computation of boundary and coboundary relations at step 3 requires time linear in the number of p-simplices, since boundary relations are constant. At step 4, the traversal of the star-graph of a simplex σ visits every arc and every node of the star-graph exactly once. Each q-simplex is in the stars of $(q+1)$ simplices of dimension $(q-1)$ and in the stars of $\frac{(q+2)\cdot(q+1)}{2}$ simplices of dimension $(q-2)$. Thus, the computation of partial relations for all $(q-2)$-simplices has time complexity proportional to $\frac{(q+2)\cdot(q+1)}{2} n_q + (q+1)n_q$, which is $O(n_q)$.

7 Retrieving Topological Relations

7.1 Retrieving Boundary Relations

All boundary relations $R_{p,p-1}$ are directly encoded in the IS data structure, while boundary relations $R_{p,q}$ with $q < p$ can be easily retrieved through relations $R_{p,p-1}$, $R_{p-1,p-2}$, …, $R_{q+1,q}$. For instance, the vertices of a tetrahedron σ, i.e., $R_{3,0}(\sigma)$, are retrieved by applying $R_{3,2}(\sigma)$, then $R_{2,1}(\tau)$, for each triangle τ in $R_{3,2}(\sigma)$, and then $R_{1,0}(\gamma)$, for each edge γ in $R_{2,1}(\tau)$.

The time complexity of this process is equal to $\Pi_{r=q+1,p+1}c$, where c is a constant. This quantity is bounded by a constant which depends on the dimension p of the simplex and on the dimension q of its faces. For instance, retrieving $R_{d,0}(\sigma)$ relation requires $O((d+1)!)$ time.

7.2 Retrieving Coboundary Relations

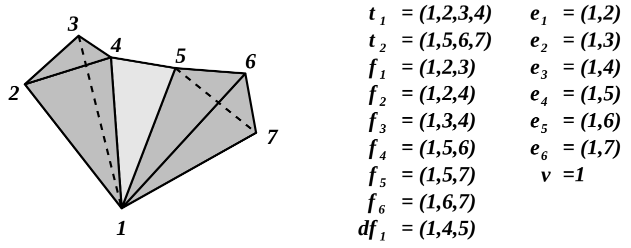
Coboundary relation $R_{d-1,d}(\sigma)$ for each $(d-1)$-simplex σ is directly encoded in the IS data structure, since $R^*_{d-1,d}(\sigma)$ is the same as $R_{d-1,d}(\sigma)$. Since only partial coboundary relations are encoded in the IS data structure, the challenge is to retrieve the complete coboundary relations efficiently.

The q-simplices incident in a p–simplex σ are either top simplices or faces of top simplices in the star of σ having dimension greater than q. Thus, in order to compute the $R_{p,q}(\sigma)$ relation, we need to retrieve the top simplices of dimensions q and above in the star of σ. This because all q-simplices that are faces of higher-dimensional simplices incident in σ can only be retrieved by considering the boundary simplices of the top simplices incident in σ.

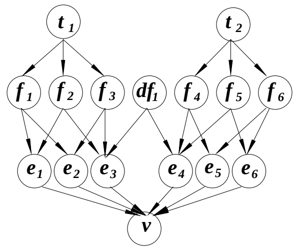
For a p-simplex σ, we consider the graph describing the topological relations encoded in the IS data structure among the simplices in the star of σ. We call this graph the *IS star-graph*. The nodes of the IS star-graph are the simplices in the star of σ, while the arcs represent the boundary and partial coboundary relations between such nodes encoded in the IS data structure. Fig. 3(a) shows an example of the star of a vertex v, while Figs. 3(b) and 3(c) show the arcs of the IS star-graph representing the boundary and the partial coboundary relations for this complex, respectively.

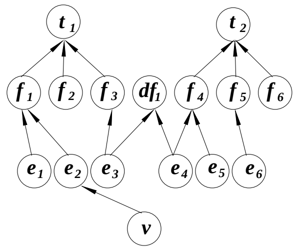
Since the retrieval algorithm performs a breadth-first traversal of the IS star-graph, each node and each arc are visited exactly once. The number of arcs is linear in the number of nodes in the IS star-graph since each simplex is bounded by a constant number of simplices. Moreover, the total number of simplices in the star of a simplex is linear in the number of top simplices in the star. Thus, the time complexity of the algorithm for retrieving coboundary relation $R_{p,q}(\sigma)$ is linear in the number of top simplices in the star of σ and, thus, it is local.

Fig. 4 shows how the traversal of the star of vertex v (from the example of Fig. 3(a)) is performed for retrieving $R_{0,1}(v)$. The traversal starts from v and it is initialized by using $R^*_{0,1}(v)$, which leads to edge e_2. Through partial



(a)





(b) (c)

Fig. 3. Example (part 1) of retrieving $R_{0,1}(v)$ through a traversal of the star of vertex v using boundary and partial coboundary relations encoded by the IS: (a) the star $st(v)$ of a vertex v; (b) boundary relations encoded by the IS among simplices in $st(v)$; (c) partial coboundary relations encoded by the IS among simplices in $st(v)$.

coboundary relations $R^*_{1,2}(e_1)$ and $R^*_{2,3}(f_1)$, tetrahedron t_1 is visited (as shown in Fig. 4(a)). Through boundary relation $R_{3,2}(t_1)$, all faces of t_1 are visited. Similarly, all the edges of faces f_1, f_2 and f_3 are visited through their boundary relations $R_{2,1}$ (see Fig. 4(b)). Partial coboundary relation $R^*_{1,2}(e_3)$ of edge e_3 leads to dangling-face df_1. Boundary relation $R_{2,1}(df_1)$ for df_1 leads to edge e_4 (as shown in Fig. 4(c)). Through edge e_4, all the faces and edges of tetrahedron t_2 are visited in a similar fashion as those of tetrahedron t_1 (see Fig. 4(d)). At the end of the traversal, all the edges that are in the coboundary relation $R_{0,1}(v)$ are retrieved.

7.3 Retrieving Adjacency Relations

Adjacency relation $R_{p,p}(\sigma)$ for a p-simplex σ with $p > 0$, is simply retrieved by first extracting all the faces τ in the boundary relation $R_{p,p-1}(\sigma)$ and then retrieving coboundary relation $R_{p-1,p}(\tau)$ for each τ. If $p = 0$, then adjacency relation $R_{0,0}(v)$ for a vertex v is obtained by first retrieving the set of edges in coboundary relation $R_{0,1}(v)$, and then retrieving the other extreme vertex of each edge e in $R_{0,1}(v)$ through boundary relation $R_{1,0}(e)$.

For $p > 0$, the running time of the algorithm for retrieving $R_{p,p}(\sigma)$ is dominated by the time required to retrieve the coboundary relations for the

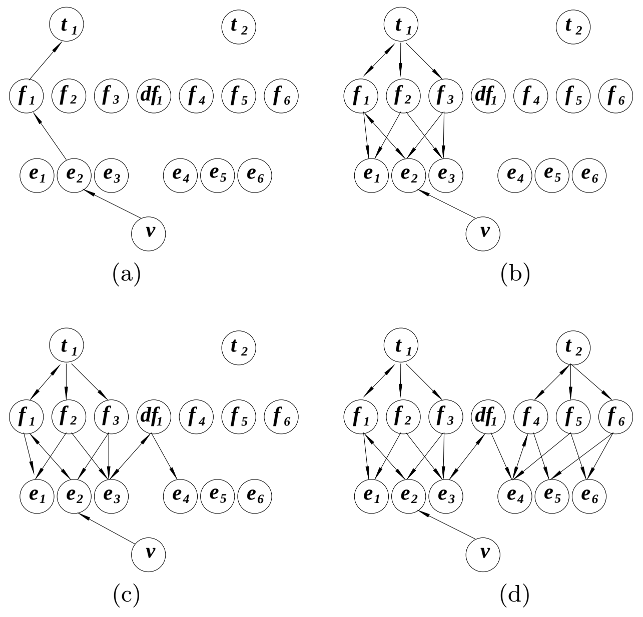


Fig. 4. Example (part 2) of retrieving $R_{0,1}(v)$ through a traversal of the star of vertex v using boundary and partial coboundary relations encoded by the IS: (a) to (d) are four stages of the traversal of $st(v)$.

$(p-1)$-faces of σ. Thus, the complexity of the algorithm is linear in the total number of top simplices incident at the $(p-1)$-faces of σ. Similarly, the time complexity of the algorithm for retrieving $R_{0,0}(v)$ is linear in the number of top simplices incident at vertex v.

8 Analysis and Comparisons

8.1 Comparison with the Incidence Graph

The *Incidence Graph* (IG) [10] stores all simplices and the same boundary relations as the IS data structure, but it encodes all coboundary relations of consecutive index $R_{p,p+1}$. In our tests, we have implemented the incidence graph for simplicial complexes: in this case, the storage cost of the IG is equal to $d+1+\sum_{0\leq p\leq d} n_p+2\sum_{0<p\leq d}(p+1)n_p$, since encoding coboundary relations $R_{p,p+1}$ requires the same space as encoding boundary relations $R_{p,p-1}$. Thus, the incidence graph occupies $\sum_{p=1}^{d-1}(p+1)n_p - \sum_{q=0}^{d-2}(\kappa_q)$ integers more than the IS data structure, where $\kappa(\sigma)$ is the total number of connected components in the link of a simplex σ and $\kappa_p = \sum_{dim(\sigma)=p}\kappa(\sigma)$, for $0 \leq p < d$ is the total number of connected components summed over the links of all p-simplices in Σ. The above difference is maximized for manifold complexes. In this case,

$\kappa_q = n_q$ and thus such difference becomes $(d+1)n_d + \sum_{q=1}^{d-1} qn_q - n_0$. Fig. 5 provides an example where the star of a vertex v is a manifold 3-complex. In this case, the incidence graph encodes all the seven edges incident at v, while the IS data structure encodes only edge e. The difference between the IG and the IS data structures is minimum when only $(q+1)$-simplices are incident at all q-simplices and in this case $\kappa_q = (q+2)n_{q+1}$. For example, if the complex is formed only by edges, then only relations $R_{0,1}$ and $R_{1,0}$ are encoded. In this case, the IG and the IS data structures have the same storage cost.

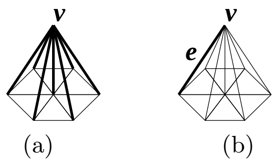


Fig. 5. A comparison between $R_{0,1}(v)$ and $R^*_{0,1}(v)$ in the manifold case: (a) the IG encodes all the edges that are incident at v, while (b) the IS encodes only edge e in the star of v.

Retrieving boundary relations is performed in the same way on the IG and on the IS data structure, and requires constant time. Retrieving coboundary relation $R_{p,q}(\sigma)$, with $p < q+1$, from the IS data structure requires time linear in the number of top simplices in the star of σ, as shown in Sect. 7.2. In the IG, coboundary relation $R_{p,r}(\gamma)$ $(p < r)$ is obtained by retrieving the encoded coboundary $R_{i,i+1}$ relations of all the i-cells for $i = p, \ldots, r-1$ in the star of γ. The retrieval of such relations can be performed in time linear in the number of involved cells. Thus, such algorithm is always optimal. In the case of simplicial 2- and 3-complexes, coboundary relations can be retrieved in optimal time from both data structures. In the case of simplicial d–complexes with $d > 3$, coboundary relations can be retrieved in optimal time only from the IG. For both the IS data structure and the IG, the time required for retrieving adjacency relations depends on the retrieval of boundary and coboundary relations. In Sect. 8.2 we present some comparisons between the IS data structure and 2D and 3D instances of the IG.

8.2 Comparison with dimension-specific data structures

In this subsection, we compare the IS data structure with dimension-specific data structures proposed for 2-dimensional and 3-dimensional simplicial complexes. For the sake of brevity, we do not compare with data structures which are specific for manifold complexes.

In the 2-dimensional case, we consider the *Directed Edge (DE)* [3] and the *Triangle-Segment (TS)* [8] data structures. The DE data structure is an edge-based data structure extending to the non-manifold case the *half-edge* data structure [18], proposed for 2-dimensional cell complexes with a manifold domain. We have shown in [7] that the DE data structure is the most space-efficient data structure among edge-based ones for encoding simplicial 2-complexes. The TS data structure is an adjacency-based data structure which extends the indexed data structure with adjacencies to arbitrary simplicial 2-complexes embedded in the 3-dimensional Euclidean space. The DE data structure encodes edges and vertices explicitly and triangles implicitly. The TS data structure encodes only the vertices and the top simplices of the complex, i.e., wire-edges and triangles. The IG and the IS data structure encode all simplices in the input complex. In this case, topological relations can be retrieved in optimal time, i.e., in time linear in the number of output simplices, from such data structures.

In [7], we have compared the DE, the TS and the 2D instance of the IG data structures on several complexes. In accordance with our results, edge-based data structures require more space than the 2D instance of the IG. For example, the DE data structure is 1.3 to 1.5 times larger than the IG. Our experiments on the same sets of complexes have shown that the IG is about 1.25 the size of the IS data structure. The specific results are not reported here for brevity.

We have also compared the IS data structure and the IG with the *Non-Manifold Indexed data structure with Adjacencies (NMIA)* [6] for simplicial 3-complexes embedded in the 3-dimensional Euclidean space. The NMIA data structure is an adjacency-based data structure which extends to the non-manifold domain the indexed data structure with adjacencies. The extension is performed by encoding the multiple connected components of the star at non-manifold vertices and non-manifold edges implicitly. The NMIA data structure encodes only vertices and top simplices in the input simplicial complex. Table 1 shows the results of the comparison with the NMIA data structure and the 3D instances of the IG and of the IS data structure. Columns n_0, n_1, n_2 and n_3 show the number of vertices, edges, faces and tetrahedra, while columns n_1^t and n_2^t provide the number of wire-edges and dangling faces in the input model. The last three columns provide the storage cost of these data structures.

Table 1. The storage costs required by the NMIA, the IG and the IS data structures in order to encode 3D non-manifold models.

Data set	n_0	n_1	n_2	n_3	n_1^t	n_2^t	NMIA	IG	IS
Bucket	53	167	160	48	6	32	591	2012	1105
Wheel	402	2093	2728	1148	96	32	10.2k	33.9k	17.7k
Ballon	1108	3913	3616	856	64	1632	13.1k	44.2k	23.4k
Flasks	1301	6307	8465	3455	0	460	30.4k	104k	53.2k
Teapot	4658	17.9k	17.0k	5666	2944	3930	73.8k	219k	120k

In this case, the NMIA data structure is the most compact one as it encodes only top simplices and vertices explicitly. Thus, it is not suitable for applications requiring explicit encoding of all simplices. The incidence graph is at least three times the size of the NMIA data structure, because it encodes all simplices and a large number of incidence relations. The IS data structure is more compact than the IG (which uses about 1.38 times as much storage as the IS) because it only encodes a subset of coboundary relations.

All topological relations can be retrieved in optimal time from the IS data structure and from the IG. Coboundary relations $R_{1,3}$ and $R_{0,3}$ can be retrieved from the NMIA data structure in time linear in the number of top simplices incident at an edge or at a vertex. Thus, they are not optimal. All the other relations can be extracted from the NMIA data structure in optimal time.

9 Concluding Remarks

We have presented the *Incidence Simplicial (IS)* data structure, a new dimension-independent data structure for representing d-dimensional simplicial complexes in the n-dimensional Euclidean space. The IS data structure has the same representation power as the widely-used *incidence graph*, but it is more compact. Furthermore, the IS data structure has the same performances in traversal and manipulation algorithms of the IG. We have presented algorithms for building the IS data structure from a soup of top simplices and for retrieving all topological relations. We have also compared the IS data structure with the incidence graph and with dimension-specific data structures, also on the basis of an experimental evaluation of their storage costs.

We are developing the *IS Library*, a robust and effective implementation of the IS data structure. This platform-independent library, written in C++, will be distributed in the public domain. Currently, the IS library contains methods for building the IS data structure from a soup of simplices, as described in Sect. 6, and methods for navigating the data structure by extracting boundary, coboundary and adjacency relations, as described in Sect. 7. All the internal data structures are already suitable for supporting update operations. We have designed and developed an implementation of the vertex-pair collapse update operator [22], which we are currently testing.

A common issue in representing and manipulating non-manifold objects is the availability of large-size simplicial representations for describing such objects. Their complexity can easily exceed the capability of computational tools for analyzing them. In these cases, adaptively simplified meshes, i.e., simplicial complexes in which the level of detail varies in different parts of the object they describe, are often required. On the other hand, accurate mesh simplification algorithms are too time consuming to be performed on-line. Thus, a multi-resolution model, which encodes the modifications performed by a simplification algorithm in a compact representation, is an effective solution. Dimension-independent non-manifold representations can be integrated with a multi-resolution framework, giving rise to a powerful tool for modeling non-manifold objects at variable resolutions.

A natural way to deal with non-manifold objects consists of decomposing them into nearly manifold components by cutting at non-manifold simplices. We are developing a decomposition algorithm based on the IS data structure and we are planning to use the resulting decomposition as the basis for performing geometric reasoning on non-manifold shapes. In particular, we are interested in computing topological invariants from the decomposition as signatures for efficient shape analysis and retrieval and in identifying non-manifold form features based on the structure of the decomposition.

Acknowledgements

This work has been partially supported by the MIUR-FIRB project SHALOM under contract number RBIN04HWR8, and by the National Science Foundation under grant CCF-0541032.

References

1. M. Agoston. *Computer Graphics and Geometric Modelling.* Springer, 2005.
2. E. Brisson. Representing geometric structures in d dimensions: topology and order. In *Proc. of the* 5^{th} *ACM Symp. on Computational Geometry*, pages 218–227. ACM Press, 1989.
3. S. Campagna, L. Kobbelt, and H.-P. Seidel. Directed Edges - a scalable representation for triangle meshes. *Jour. of Graphics Tools*, 3(4):1–12, 1998.
4. A. Castelo, H. Lopes, and G. Tavares. Handlebody representation for surfaces and Morse operators. In J. Warren, editor, *Proc. on Curves and Surfaces for Computer Vision and Graphics III, SPIE*, pages 270–283, Boston, 1992.
5. L. De Floriani, D. Greenfieldboyce, and A. Hui. A data structure for non-manifold simplicial d-complexes. In L. Kobbelt, P. Schroder, and H. Hoppe, editors, *Proc. of the* 2^{nd} *Eurographics Symp. on Geometry Processing*, pages 83–92, Nice, France, 8–10 July 2004.
6. L. De Floriani and A. Hui. A scalable data structure for three-dimensional non-manifold objects. In L. Kobbelt, P. Schroder, and H. Hoppe, editors, *Proc. of the* 1^{st} *Eurographics Symp. on Geometry Processing*, pages 72–82, Aachen, Germany, 23–25 June 2003.
7. L. De Floriani and A. Hui. Data structures for simplicial complexes: an analysis and a comparison. In M. Desbrun and H. Pottmann, editors, *Proc. of the* 3^{rd} *Eurographics Symp. on Geometry Processing*, pages 119–128, Vienna, Austria, 4–6 July 2005.
8. L. De Floriani, P. Magillo, E. Puppo, and D. Sobrero. A multi-resolution topological representation for non-manifold meshes. *CAD Journal*, 36(2):141–159, February 2004.
9. D. Dobkin and M. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica*, 5(4):3–32, 1989.

10. H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer, 1987.
11. T. Gurung and J. Rossignac. SOT: a compact representation for tetrahedral meshes. In *Proc. of the SIAM/ACM Joint Conference on Geometric and Physical Modeling*, pages 79–88, San Francisco, USA, 2009.
12. A. Hui, L Vaczlavik, and L. De Floriani. A decomposition-based representation for 3D simplicial complexes. In *Proc. of the* 4^{th} *Eurographics Symp. on Geometry Processing*, pages 101–110, Cagliari, Italy, June 2006.
13. M. Kallmann and D. Thalmann. Star Vertices: a compact representation for planar meshes with adjacency information. *Jour. of Graphics Tools*, 6(1):7–18, 2001.
14. M. Lage, T. Lewiner, H. Lopes, and L. Velho. CHF: a scalable topological data structure for tetrahedral meshes. In *Proc. of the* 18^{th} *Brazilian Symp. on Computer Graphics and Image Processing*, pages 349–356, 2005.
15. S. H. Lee and K. Lee. Partial-entity structure: a fast and compact non-manifold boundary representation based on partial topological entities. In *Proc. of the* 6^{th} *ACM Symp. on Solid Modeling and Applications*, pages 159–170, Ann Arbor, USA, June 2001. ACM Press.
16. P. Lienhardt. Topological models for boundary representation: a comparison with n-dimensional generalized maps. *CAD Journal*, 23(1):59–82, 1991.
17. H. Lopes and G. Tavares. Structural operators for modeling 3-manifolds. In *Proc. of the* 4^{th} *ACM Symp. on Solid Modeling and Applications*, pages 10–18. ACM Press, May 1997.
18. M. Mantyla. *An Introduction to Solid Modeling.* Computer Science Press, 1987.
19. S. McMains. *Geometric Algorithms and Data Representation for Solid Freeform Fabrication.* PhD thesis, University of California at Berkeley, 2000.
20. A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-independent modeling with simplicial complexes. *ACM Trans. on Graphics*, 12(1):56–102, 1993.
21. S. Pesco, G. Tavares, and H. Lopes. A stratification approach for modeling two-dimensional cell complexes. *Computers and Graphics*, 28:235–247, 2004.
22. J. Popovic and H. Hoppe. Progressive simplicial complexes. In *Proc. of the ACM Computer Graphics*, pages 217–224. ACM Press, 1997.
23. J. Rossignac, A. Safonova, and A. Szymczak. 3D compression made simple: Edge-Breaker on a Corner table. In *Proc. of the Shape Modeling International*, Genova, Italy, May 2001. IEEE Computer Society.