Discrete Comput Geom (2009) 41: 96–118
DOI 10.1007/s00454-008-9073-y

Coreduction Homology Algorithm

Marian Mrozek · Bogdan Batko

Received: 6 June 2007 / Revised: 23 January 2008 / Accepted: 2 March 2008 /
Published online: 8 April 2008
© Springer Science+Business Media, LLC 2008

Abstract This paper presents a new reduction algorithm for the efficient computation of the homology of cubical sets and polotypes. The algorithm—particularly strong for low-dimensional sets embedded in high dimensions—runs in linear time. The paper presents the theoretical background of the algorithm, the algorithm itself, experimental results based on an implementation for cubical sets as well as some theoretical complexity estimates.

Keywords Homology algorithm · Cubical set · Reduction methods

1 Introduction

This paper presents a new reduction algorithm for preprocessing homology computations of large cubical or simplicial complexes. The algorithm is based on the concept of one space homology theory, which enables the dual process of coreductions. The complexity of the algorithm is linear both in the size and the dimension of the input. Experimental results based on an implementation for cubical sets show that the algorithm performs much better than other available homology algorithms not only for

Both authors are partially supported by Polish MNSzW, Grant N201 037 31/3151.

M. Mrozek (✉)
Institute of Computer Science, Jagiellonian University, ul. Nawojki 11, 30-072 Kraków, Poland
e-mail: Marian.Mrozek@ii.uj.edu.pl

M. Mrozek · B. Batko
Division of Computational Mathematics, Graduate School of Business, ul. Zielona 27,
33-300 Nowy Sącz, Poland

B. Batko
Institute of Mathematics, Pedagogical University, ul. Podchorążych 2, 30-084 Kraków, Poland
e-mail: bbatko@ap.krakow.pl

cubical sets with simple topology (sphere, torus, Bing's house, Klein bottle etc.) but also for randomly generated cubical sets. Complexity estimates concerning rescalings of a fixed cubical set indicate that the algorithm is better than other homology algorithms particularly for low-dimensional sets regardless of the embedding dimension.

The classical homology algorithm is based on Smith diagonalization of the matrix of the boundary homomorphism [22, Sect. 1.11]. The computational complexity of the best available Smith diagonalization algorithm is $O(n^{3.376\dots})$ [24]. Several authors propose variants or alternatives for the classical approach [1, 7–9, 27], but only Delfinado and Edelsbrunner [3] presented an algorithm for Betti numbers that runs in near linear time. Unfortunately, the applicability of this algorithm is restricted to dimension three. Donald and Chang [4] performed a probabilistic analysis of the Smith diagonalization for a certain class of sparse matrices with invertible entries and showed that the expected complexity of homology algorithm is quadratic in the number of generators if the matrices of the boundary maps are in this class. However, it does not seem likely that this bound can be improved on the basis of the sparseness of boundary maps only, because of the well-known fill-in process [5]. For instance, applying the Smith algorithm to diagonalize the following sparse matrix

$$\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \dots \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & \dots \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & \dots \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & \dots \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & \dots \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & \dots \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \dots \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots \\
. & . & . & . & . & . & . & . & \dots \\
. & . & . & . & . & . & . & . & \dots
\end{bmatrix}$$

we observe the fill-in process in columns. One can try to minimize the fill-in process by searching for the best order of the operations on matrices. However, unlike the Gauss algorithm, the Smith algorithm does not leave much room for changing the order. Moreover, finding an ordering that minimizes the fill-in is NP-complete (see [26]).

The quadratic expected complexity obtained by Donald and Chang is significantly better than the general complexity of the Smith diagonalization, but chain complexes in which the number of generators exceeds 10^5 are on the verge of the applicability of the quadratic algorithms. This is unsatisfactory for applications in rigorous numerics of dynamical systems and in image analysis by topological methods (see [12, 20] for a broader discussion of this subject). Moreover, in some applications, particularly in image analysis it is important to perform homology computations in real time. Clearly such applications require new ideas. Some authors search for improvements to the Smith algorithm, in particular by applying probabilistic methods [6, 7]. Tests show that this approach is helpful only to cope with many large torsion numbers. Moreover, such methods cannot be used in rigorous numerics.

However, finding homology is a problem in computational topology and not in general computational algebra, therefore one can hope for some methods specific to the problem to be fruitful. The methods of chain complex reduction, originally proposed in [11] and then developed in [12, 14, 17], constitute such an approach. They consist in iterating the process of replacing the chain complex, or even better some combinatorial representation of the topological space, by a smaller one with the same homology and computing the homology only when no more reductions are possible. This way one postpones the process of computing the homology of the chain complex until the complex is acceptably small. Moreover, if the reduction process is applied directly to the combinatorial representation of the topological space, then one also postpones the expensive process of constructing the chain complex until the space is small. Of course, one can profit from the reduction process only if one step of the reduction is computationally inexpensive and the reduction is significant.

The algorithm presented in [11] is a modification of Smith diagonalization for sparse matrices. The modification consists in removing columns and rows of generators with trivial homology in all relevant matrices of boundary maps as soon as the triviality is verified. This simple modification helps to keep the fill-in process at a moderate level and numerical experiments show that the expected complexity of this algorithm is significantly below quadratic.

The most expensive part of one step of the Smith diagonalization, in the case of an invertible pivot element, is the cost of modifying the matrix. The cost is $O((p-1)(q-1))$, where p and q denote the number of nonzero elements, respectively, in the row and column of the pivot element. There are two special cases when this cost is zero: the case of an elementary reduction, when the pivot entry is the unique nonzero element in its row, and the dual case of an elementary coreduction, when the pivot entry is the unique nonzero element in its column. As long as such special cases are available there is no need to store the matrices of boundary maps if only a combinatorial representation of the topological space is available. Obviously, these types of reductions are preferable, because they may be performed very fast.

On the topological level, the case of an elementary reduction corresponds to a deformation retraction of a free face onto the complex. The problem is that in most situations free face reductions are quickly exhausted, so the benefit is short lasting.

An elementary coreduction is not possible in a standard simplicial or cubical complex, because the boundary of a vertex is always zero and there are at least two elements in the boundary of all other simplices or cubes. In this paper we show how one can benefit from the elementary coreductions in homology computations. This is done by means of the one space homology theory, i.e. the homology theory which does not require relative homology for building the long exact sequence of a pair of topological spaces (see [16, 23]).

Since an elementary reduction or coreduction takes only constant time, the complexity of an algorithm that runs through all generators and performs reductions and/or coreductions whenever possible is linear. Therefore the complexity of a homology algorithm preprocessed by elementary reductions or coreductions depends on the size of the complex resulting from the preprocessing. Numerical experiments as well as some partial complexity results show that unlike the case of elementary reductions, the elementary coreductions may be performed very deeply, which results in a very fast homology algorithm.

The algorithm presented in this paper may be viewed as a special form of the acyclic subspace homology algorithm presented in [20]. The acyclic subspace homology algorithm so far outperforms other available homology software. Unfortunately, it significantly slows down when the embedding dimension of the set is increased. This is because the construction of the acyclic subspace proposed there is based on the technique of intersection with neighbours. This requires testing all neighbours ($3^d - 1$ neighbours in the case of a d-dimensional cube), which means that the complexity of the algorithm depends exponentially on the dimension of the space. On the other hand, the coreduction technique proposed in this paper may be viewed as the construction of an acyclic subspace which requires only testing the codimension one neighbours ($2d$ neighbours in the case of a d-dimensional cube).

The first, purely topological approach to the combinatorial idea of the coreduction introduced in this paper is presented in [21]. It is restricted to cubical sets only and it is based on the local compactness criterion for representable sets instead of the more general concept of regular subsets of S-complexes introduced in this paper.

The organization of the paper is as follows. In Sect. 2 we introduce the concept of an S-complex. In Sect. 3 we present a combinatorial approach to the one space homology theory of S-complexes. The concept of coreductions is introduced in Sect. 4. An example is discussed in Sect. 5. The algorithm is presented in Sect. 6. In Sect. 7 we show the results of some numerical experiments and in the last section we present a theoretical result concerning the complexity of the algorithm under rescalings.

2 *S*-complexes

Recall that a sequence $(X_q)_{q\in\mathbb{Z}}$ of objects of a category $\mathcal{C}$ is a gradation of an object X in $\mathcal{C}$ if X decomposes as the direct sum of X_q. In particular, in the case of the category of sets the gradation is the decomposition into a disjoint union and in the category of moduli the gradation is the decomposition into the algebraic direct sum.

Let R be a ring with unity. Given a finite set A let $R(A)$ denote the free module over R generated by A.

Let S be a finite set with a gradation S_q such that $S_q = \emptyset$ for $q < 0$. Then $R(S_q)$ is a gradation of $R(S)$ in the category of moduli over the ring R. For every element $s \in S$ there exists a unique number q such that $s \in S_q$. This number will be referred to as the *dimension* of s and denoted $\dim s$.

We use the notation $\langle\cdot,\cdot\rangle : R(S) \times R(S) \to R$ for the scalar product defined on generators by

$$\langle t, s\rangle = \begin{cases} 1, & t = s, \\ 0, & \text{otherwise} \end{cases}$$

and extended bilinearly to $R(S) \times R(S)$.

Let $\kappa : S \times S \to R$ be a map satisfying

$$\kappa(s,t) \neq 0 \quad\Longrightarrow\quad \dim s = \dim t + 1.$$

We say that (S,κ) is an S-complex if $(R(S),\partial^\kappa)$ with $\partial^\kappa : R(S) \to R(S)$ defined on generators $s \in S$ by

$$\partial^\kappa(s) := \sum_{t\in S} \kappa(s,t)t$$

is a free chain complex with base S. The map κ will be referred to as the *coincidence index*. If $\kappa(s,t) \neq 0$, then we say that t is a *face* of s and S is a *coface* of t.

By the homology of an S-complex (S,κ) we mean the homology of the chain complex $(R(S),\partial^\kappa)$ and we denote it by $H(S,\kappa) := H(R(S),\partial^\kappa)$ or simply by $H(S)$. In the following we will drop the superscript κ in ∂^κ whenever κ is clear from the context.

Of course, an S-complex is only a reformulation of a chain complex, because the map κ may be viewed as the matrix of the boundary homomorphism. By referring to an S-complex instead of a chain complex we want only to stress that in many applications the natural coding of S carries in itself the information about κ, so there is no need at all to store κ in memory throughout the process of reductions and the reductions may be performed on the coding of S only. The two main examples of such S-complexes are simplicial complexes and cubical complexes.

Recall that a *q-simplex* $\sigma = [A_0, A_1, \dots, A_q]$ in $\mathbb{R}^d$ is the convex hull of $q+1$ affine independent points $A_0, A_1, \dots, A_q$ in $\mathbb{R}^d$, called the *vertices* of σ. The number q is the *dimension* of the simplex. A *face* of σ is a simplex whose vertices constitute a subset of $(A_0, A_1, \dots, A_q)$. A *simplicial complex* consists of a collection $\mathcal{S}$ of simplices such that every face of a simplex in $\mathcal{S}$ is in $\mathcal{S}$ and the intersection of two simplices in $\mathcal{S}$ is their common face. The simplicial complex $\mathcal{S}$ has a natural gradation $(\mathcal{S}_q)$, where $\mathcal{S}_q$ consists of simplices of dimension q. Since a zero-dimensional simplex is the singleton of its unique vertex, $\mathcal{S}_0$ may be identified with the collection of all vertices of all simplices in the simplicial complex $\mathcal{S}$.

Assume an ordering of $\mathcal{S}_0$ is given and every simplex σ in $\mathcal{S}$ is coded as $[A_0, A_1, \dots, A_q]$, where the vertices $A_0, A_1, \dots, A_q$ are listed according to the prescribed ordering of $\mathcal{S}_0$. By putting

$$\kappa(\sigma,\tau) := \begin{cases} (-1)^i, & \text{if } \sigma = [A_0, A_1, \dots, A_q] \\ & \text{and } \tau = [A_0, A_1, \dots, A_{i-1}, A_{i+1}, \dots, A_q], \\ 0, & \text{otherwise,} \end{cases}$$

we obtain an S-complex whose chain complex is the classical simplicial chain complex used in simplicial homology.

Let $I = [k,l] \subset \mathbb{R}$ be a compact interval. Its *length* is defined as $l-k$ and denoted len I. The interval I is called an *elementary interval* if len $I \in \{0,1\}$ and its endpoints are integers. Elementary intervals of length one are called *nondegenerate*. We define the *left interval* of I by $I^- := [k,k]$ and the *right interval* of I by $I^+ := [l,l]$.

An *elementary cube* in $\mathbb{R}^d$ is the Cartesian product $Q = I_1 \times \cdots \times I_d$ of d elementary intervals. The *dimension* of Q is the number of nondegenerate intervals in the product decomposition. A *full* elementary cube is an elementary cube whose product decomposition consists only of nondegenerate intervals. For every elementary cube

Q and number $j \in \{1, 2, \dots, d\}$ we define the jth *nondegeneracy number* of Q by

$$\nu(Q, j) := \begin{cases} \operatorname{card}\{i < j \mid \operatorname{len} I_i = 1\}, & \text{if } \operatorname{len} I_j = 1, \\ 0, & \text{otherwise.} \end{cases}$$

A *cubical complex* $\mathcal{C}$ in $\mathbb{R}^d$ is a finite collection of elementary cubes in $\mathbb{R}^d$ and the associated *cubical set* is the union of this collection. The cubical complex $\mathcal{C}$ has a natural gradation $(\mathcal{C}_q)_{q \in \mathbb{Z}}$, where $\mathcal{C}_q$ consists of elementary cubes of dimension q. We put

$$\kappa(Q, P) := \begin{cases} (-1)^{\nu(Q,j)}, & \text{if } Q = I_1 \times \cdots \times I_j \times \cdots \times I_d \\ & \text{and } P = I_1 \times \cdots \times I_j^- \times \cdots \times I_d, \\ (-1)^{1+\nu(Q,j)}, & \text{if } Q = I_1 \times \cdots \times I_j \times \cdots \times I_d \\ & \text{and } P = I_1 \times \cdots \times I_j^+ \times \cdots \times I_d, \\ 0, & \text{otherwise.} \end{cases}$$

An induction argument in d may be used to show that a cubical complex is an S-complex. The associated chain complex coincides with the cubical chain complex used in [12] to define the homology of *cubical sets*, i.e. finite unions of elementary cubes.

In both of these examples the internal structure of the generators carries all the information needed to compute the coincidence index of two generators in constant time. Therefore such a complex may be represented in memory by storing the generators only: there is no need to store the coincidence index. In the case of a cubical complex the memory representation may be taken to be a bitmap, which is particularly compact.

3 Regular Subsets of S-complexes

We want to replace the set of generators S by a subset $S' \subset S$ which, via the coincidence index $\kappa' := \kappa|_{S' \times S'}$, gives rise to another S-complex with the same homology. Note that we take κ' as the restriction of κ to guarantee that κ' may be recovered directly from the coding of S'.

The first question to address is what is a sufficient condition for (S', κ') to be an S-complex. To answer this question let us introduce the following notation. Given $A \subset S$ put

$$\operatorname{bd}_S A := \{t \in S \mid \kappa(s, t) \neq 0 \text{ for some } s \in A\},$$
$$\operatorname{cbd}_S A := \{s \in S \mid \kappa(s, t) \neq 0 \text{ for some } t \in A\}.$$

In the following we drop the braces in $\operatorname{bd}_S\{s\}$ or $\operatorname{cbd}_S\{s\}$ and write $\operatorname{bd}_S s$ or $\operatorname{cbd}_S s$ when applying this notation to a singleton of an $s \in S$.

We have the following theorem

Theorem 3.1 *If $S' \subset S$ is such that for all $s, u \in S'$ and $t \in S$*

$$t \in \mathrm{bd}_S\, s \quad \textit{and} \quad u \in \mathrm{bd}_S\, t \quad \textit{implies } t \in S', \tag{1}$$

then (S', κ') is an S-complex.

Proof We need to verify that $\partial^{\kappa'}\partial^{\kappa'} = 0$. Assume this is not true. Then

$$0 \neq \partial^{\kappa'}\partial^{\kappa'}(s) = \sum_{t\in S'} \kappa(s,t)\partial^{\kappa'} t = \sum_{u\in S'}\left(\sum_{t\in S'}\kappa(s,t)\kappa(t,u)\right)u$$

for some $s \in S'$. Therefore there exists a $u_0 \in S'$ such that

$$\sum_{t\in S'} \kappa(s,t)\kappa(t,u_0) \neq 0. \tag{2}$$

On the other hand ∂^{κ} is a boundary, so we have

$$0 = \partial^{\kappa}\partial^{\kappa}(s) = \sum_{t\in S} \kappa(s,t)\partial^{\kappa} t = \sum_{u\in S}\left(\sum_{t\in S}\kappa(s,t)\kappa(t,u)\right)u.$$

In particular

$$\sum_{t\in S} \kappa(s,t)\kappa(t,u_0) = 0.$$

It follows from (2) that

$$\sum_{t\in S\setminus S'} \kappa(s,t)\kappa(t,u_0) \neq 0,$$

which implies that $\kappa(s,t_0)\kappa(t_0,u_0) \neq 0$ for some $t_0 \in S \setminus S'$. Therefore $t_0 \in \mathrm{bd}\, s$, $u_0 \in \mathrm{bd}\, t_0$ and $t_0 \notin S'$, a contradiction. □

We say that $S' \subset S$ satisfying (1) is *regular*. We say that $S' \subset S$ is *closed* in S if $\mathrm{bd}_S\, S' \subset S'$. We say that $S' \subset S$ is *open* in S if $S \setminus S'$ is closed in S.

Theorem 3.2 *If $S' \subset S$ is closed in S, then S' and $S \setminus S'$ are regular.*

Proof Let S' be closed in S. The fact that S' is regular is obvious. Assume $S \setminus S'$ is not regular. Then there exist $s, u \in S \setminus S'$ and $t \in S \setminus (S \setminus S')$ such that $\kappa(s,t) \neq 0 \neq \kappa(t,u)$. Thus $t \in S'$ and since S' is closed we get $u \in S'$, a contradiction. □

Theorem 3.3 *If S' is closed in S, then*

(i) $\partial^{\kappa'} = \partial^{\kappa}|_{R(S')}$
(ii) *$R(S')$ is a subcomplex of $R(S)$.*

Proof To show (i) observe that since $\operatorname{bd} t \subset S'$ for $t \in S'$, we have

$$\partial^\kappa(t) = \sum_{u \in S} \kappa(t,u)u = \sum_{u \in S \cap \operatorname{bd} t} \kappa(t,u)u = \sum_{u \in S' \cap \operatorname{bd} t} \kappa(t,u)u = \partial^{\kappa'}(t).$$

Now $\partial^\kappa(t) = \partial^{\kappa'}(t) \in R(S')$ for every $t \in S'$. Therefore $\partial^\kappa(R(S')) \subset R(S')$, which proves (ii). □

However, let us remark that the conclusion of Theorem 3.3 need not be true when S' is not closed in S.

Theorem 3.4 *Assume $S' \subset S$ is closed in S. Let $S'' := S \setminus S'$ and $\kappa'' := \kappa|_{S'' \times S''}$. Then the inclusion*

$$\iota : \bigl(R(S'), \partial^{\kappa'}\bigr) \to \bigl(R(S), \partial^\kappa\bigr),$$

and the projection

$$\pi : \bigl(R(S), \partial^\kappa\bigr) \to \bigl(R(S''), \partial^{\kappa''}\bigr)$$

are chain maps. Moreover, we have the following short exact sequence

$$0 \to R(S') \xrightarrow{\iota} R(S) \xrightarrow{\pi} R(S'') \to 0 \tag{3}$$

and the following long exact sequence of homology modules

$$\cdots \xrightarrow{\partial_{k+1}} H_q(S') \xrightarrow{\iota_q} H_q(S) \xrightarrow{\pi_q} H_q(S'') \xrightarrow{\partial_k} H_{q-1}(S') \xrightarrow{\iota_{q-1}} \cdots. \tag{4}$$

Proof The fact that ι is a chain map follows immediately from Theorem 3.3. To show that π is a chain map take $s \in S$ and consider first the case $s \in S'$. Then $\partial^\kappa(s) \in R(S')$ and

$$\pi\bigl(\partial^\kappa(s)\bigr) = 0 = \partial^{\kappa''}(0) = \partial^{\kappa''}\bigl(\pi(s)\bigr).$$

On the other hand, if $s \in S''$, then

$$\pi\bigl(\partial^\kappa(s)\bigr) = \pi\left(\sum_{t \in S} \kappa(s,t)t\right) = \sum_{t \in S''} \kappa(s,t)t = \partial^{\kappa''}(s) = \partial^{\kappa''}\bigl(\pi(s)\bigr)$$

which proves that π is a chain map.

The fact that the sequence (3) is exact is obvious and the exactness of the sequence (4) follows from the standard result in homological algebra. □

Theorem 3.5 *If S' is closed in S, then*

$$H(S'') \cong H\bigl(R(S), R(S')\bigr).$$

Proof The projection π induces a chain map $\bar{\pi}: R(S)/R(S') \to R(S'')$. It is straightforward to check that the following diagram, in which the top row is the exact sequence of a pair and the unmarked vertical arrows denote identities, is commutative.

$$\begin{tikzcd}
\cdots \arrow[r, "\partial_{k+1}"] & H_q(S') \arrow[r, "\iota_q"] \arrow[d] & H_q(S) \arrow[r, "\pi_q"] \arrow[d] & H_q(R(S), R(S')) \arrow[r, "\partial_k"] \arrow[d, "\bar{\pi}"] & \cdots \\
\cdots \arrow[r, "\partial_{k+1}"] & H_q(S') \arrow[r, "\iota_q"] & H_q(S) \arrow[r, "\pi_q"] & H_q(S'') \arrow[r, "\partial_k"] & \cdots
\end{tikzcd}$$

Now the conclusion follows from the five lemma. □

We say that a regular subset $T \subset S$ is a *null set* of S if T is closed or open in S and $H(R(T)) = 0$. As an immediate consequence of Theorem 3.4 we get the following corollary.

Corollary 3.6 *If T is a null set of S, then $H(R(S))$ and $H(R(S \setminus T))$ are isomorphic.*

The corollary tells us that if we are able to locate a null set in S, then we can remove it without changing the homology of S. In the next section we will indicate a simple method of locating null sets in S.

We conclude this section with the following lemma, which will be useful later.

Lemma 3.7 *Assume $s_0, t_0, u_0 \in S$ are such that $u_0 \in \operatorname{bd} t_0$ and $t_0 \in \operatorname{bd} s_0$. Then $\operatorname{card} \operatorname{bd} s_0 \geq 2$ and $\operatorname{card} \operatorname{cbd} u_0 \geq 2$.*

Proof We have

$$0 = \partial\partial s_0 = \sum_{u \in S} \left(\sum_{t \in S} \kappa(s_0, t)\kappa(t, u) \right) u.$$

In particular

$$0 = \sum_{t \in S} \kappa(s_0, t)\kappa(t, u_0).$$

Since by our assumptions the term $\kappa(s_0, t_0)\kappa(t_0, u_0) \neq 0$, the sum may be zero only if there is another nonzero term. Thus

$$\kappa(s_0, t_1) \neq 0 \neq \kappa(t_1, u_0) \quad \text{for some } t_1 \neq t_0 \in S.$$

It follows that $t_0, t_1 \in \operatorname{bd} s_0$ and $t_0, t_1 \in \operatorname{cbd} u_0$. □

4 Reduction Pairs

As in [11] we say that a pair (a, b) of elements of S is a reduction pair if $\kappa(b, a)$ is invertible in R. A reduction pair (a, b) is said to be an *elementary reduction pair* if

$\operatorname{cbd}_S a = \{b\}$. In this case we will also say that a is a *free face* in S. Similarly, we define an *elementary coreduction pair* as a reduction pair (a, b) such that $\operatorname{bd}_S b = \{a\}$ and in this case we call b a *free coface* in S.

Theorem 4.1 *Assume $a, b \in S$. If (a, b) is an elementary reduction pair, then $\{a, b\}$ is open in S. If (a, b) is an elementary coreduction pair, then $\{a, b\}$ is closed in S. Moreover, in both cases $\{a, b\}$ is a null set.*

Proof First assume that (a, b) is an elementary reduction pair. We need to show that $S \setminus \{a, b\}$ is closed. Assume this is not true. Then there exists an $s_0 \in S \setminus \{a, b\}$ and a $t_0 \in \operatorname{bd} s_0 \cap \{a, b\}$. It cannot be $t_0 = a$, because then $b \neq s_0 \in \operatorname{cbd}_S a$. Therefore $t_0 = b$. By Lemma 3.7 $\operatorname{card}\operatorname{cbd}_S a \geq 2$, a contradiction.

Next assume that (a, b) is an elementary coreduction pair. We need to show that $\{a, b\}$ is closed. Obviously $\operatorname{bd}_S a = \{b\} \subset \{a, b\}$. We also have $\operatorname{bd}_S a = \emptyset$, because otherwise Lemma 3.7 implies that $\operatorname{card}\operatorname{bd}_S b \geq 2$. Therefore $\operatorname{bd}_S\{a, b\} \subset \{a, b\}$.

Let $T := \{a, b\}$. Let $k := \dim b$. Then

$$R_q(T) = \begin{cases} Ra & \text{for } q = k-1, \\ Rb & \text{for } q = k, \\ 0, & \text{otherwise,} \end{cases}$$

and ∂_q is zero except ∂_k, which is the multiplication by $\kappa(b, a)$. Since $\kappa(b, a)$ is invertible, ∂_k is an isomorphism. It follows that $Z_q(T) = 0$ for all $q \neq k-1$ and $Z_{k-1}(T) = Ra = B_{k-1}(T)$. Therefore $H_q(T) = 0$. □

Thus, Theorem 4.1 and Corollary 3.6 imply the following corollary.

Corollary 4.2 *If (a, b) is an elementary reduction or coreduction pair in S, then $H(R(S))$ and $H(R(S \setminus \{a, b\}))$ are isomorphic.*

5 An Example

The simplest elementary reduction algorithm consists in running through all the generators and performing the reduction whenever a reduction or coreduction pair is found. This is illustrated in Fig. 1. We begin with a simplicial complex consisting of one triangle BCD, eight edges AB, AD, AE, AF, BC, BD, CD, DE and six vertices A, B, C, D, E, F. Let us assume that the simplices are analysed in the lexicographic order

$$A, AB, AD, AE, AF, B, BC, BD, BCD, C, CD, D, DE, E, F.$$

Then the consecutive elementary reduction pairs encountered are

$$(A, AF), \quad (BC, BCD) \quad \text{and} \quad (C, CD)$$

and the resulting complex consists of five edges AB, AD, AE, BD, DE and four vertices A, B, D, E. However, there are at least two problems with this algorithm:

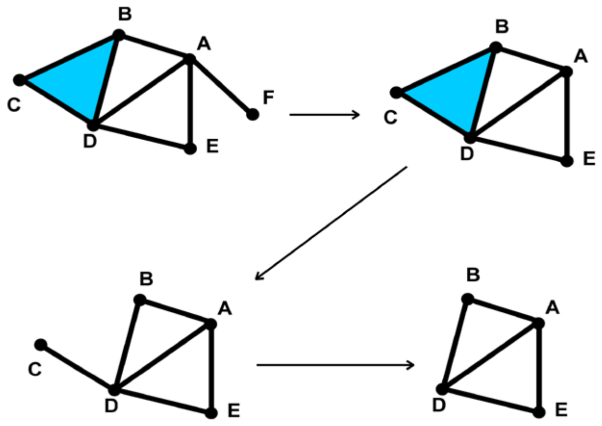

Fig. 1 Simple reduction

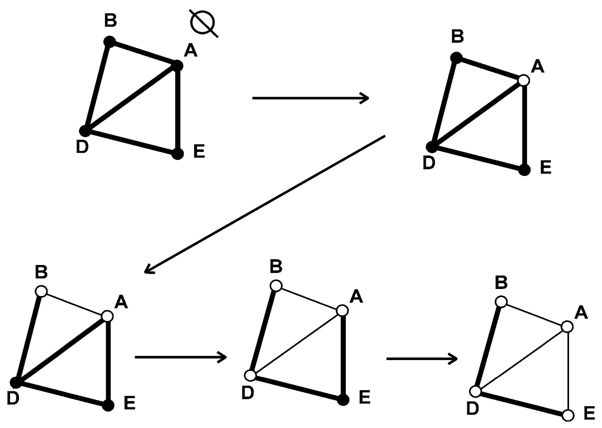

Fig. 2 Simple coreduction. The missing vertices are marked with *circles* and the missing edges are marked with *thin lines*

the depth of the reduction crucially depends on the order in which the generators are analysed and the output of the algorithm cannot be expected to be small if the minimal homology generators are large.

Consider now the outcome of the reduction algorithm applied to the considered example from the point of view of possible coreductions. First recall that no simplicial complex may admit an elementary coreduction pair, because the cardinality of boundary is $n+1$ in the case of an n-dimensional simplex with $n>0$ and zero for a vertex, so it is never one. However, when we treat the simplicial complex as the reduced complex, i.e. we assume that the empty set is an additional simplex of dimension -1 which has in its coboundary all vertices, then the situation becomes different, because every vertex becomes a free coface with the empty set as the unique element in the coboundary. The resulting sequence of coreductions is presented in Fig. 2. The outcome is an S-complex consisting of exactly two generators of dimension one, whose boundary is zero. Therefore these generators are also homology generators and obviously there are no more homology generators.

Note that if the simplicial complex has more than one connected component, then the above procedure works only for the connected component of the vertex which was picked up as a free coface. This is because once this coface is reduced, the empty set is gone and there are no more coreductions of this type available for the remaining components. A straightforward remedy is to add one extra generator in dimension -1 per each connected component with its coboundary consisting of all vertices in this component.

Observe that testing if (a, b) is an elementary reduction or coreduction pair requires only the knowledge of $\operatorname{bd} b$ or $\operatorname{cbd} a$. In all cubical complexes and many simplicial complexes the size of the boundaries and coboundaries is bounded by a small number depending on the dimension of the space. In this case the test may be performed in constant time. What is more important, the bound remains valid for the new, reduced complex. Therefore the algorithm runs in linear time. On the algebraic level this fact may be considered as the no fill-in process in the matrix of the boundary operator.

6 Coreduction Homology Algorithm

The order in which the generators are processed matters, because every reduction performed may give birth to new reduction or coreduction pairs. Therefore Algorithm 6.1 uses a method of queuing the neighbours of the last reduced pair to minimize the effect of the order on the depth of reduction without sacrificing its speed.

Algorithm 6.1 (Coreduction)

```
function Coreduction ($S$-complex $S$, a generator $s$)
begin
  $Q :=$ empty queue of generators;
  enqueue($Q, s$);
  while $Q \neq \emptyset$ do begin
    $s :=$ dequeue($Q$);
    if $\operatorname{bd}_S s$ contains exactly one element $t$ then begin
      $S := S \setminus \{s\}$;
      foreach $u \in \operatorname{cbd}_S t$ do
        if $u \notin Q$ then enqueue($Q, u$);
      $S := S \setminus \{t\}$;
    end
    else if $\operatorname{bd}_S s = \emptyset$ then
      foreach $u \in \operatorname{cbd}_S s$ do
        if $u \notin Q$ then enqueue($Q, u$);
  end;
  return $S$;
end;
```

As we commented in Sect. 5, the above form of the algorithm should be applied only to connected sets and in the general case the algorithm should be applied independently to each connected component of the set. In the actual implementation it is not difficult to combine the coreduction algorithm with the separation to connected components, so that the two algorithms work in tandem and there is no need to scan the input twice.

Also let us comment that in the actual implementation it is preferable to use two independent data structures to represent Q: a set and a queue, because apart from the standard queue operations we also verify whether a given cube is already contained in the queue or not, which cannot be done efficiently for queues.

Theorem 6.2 *Algorithm 6.1 called with an S-complex S returns a subset S' of S such that $H(S) \cong H(S')$. Moreover, if S is such that for some $M > 0$ and for every $s \in S$ there are at most M elements in $\mathrm{bd}_S\, s$, then the "`while`" loop is passed at most $2M \operatorname{card} S$ times.*

Proof Put

$$K := \{k \in \mathbb{N} \mid \text{the "}\mathtt{while}\text{" loop is passed at least } k \text{ times}\}.$$

For $k \in K$ let S_k, s_k and t_k denote, respectively, the contents of variable S, s and t on entering the kth pass of the "`while`" loop. If $k-1, k \in K$, then either $S_k = S_{k-1}$ or $S_k = S_{k-1} \setminus \{s_k, t_k\}$. The latter case happens when (s_k, t_k) is a coreduction pair. It follows from Theorem 4.1 and Corollary 3.6 that $H(S_k) \cong H(S_{k+1})$.

Observe that if an element $t \in S$ contributes its coboundary elements to Q as an element of a coreduction pair (s, t), then it is removed from S, so it may never again contribute in this role. Similarly, if an element s with zero boundary contributes its coboundary elements to Q, it may never again do it in this role, because it is not in the coboundary of any element, so it may not reappear in Q. Therefore each element of S may appear in Q at most $2M$ times. It follows that the "`while`" loop is passed at most $2M \operatorname{card} S$ times. □

As an immediate corollary we get

Corollary 6.3 *Assume that for some constant $M > 0$ and for every $s \in S$ the cardinalities of $\mathrm{bd}_S\, s$ and $\mathrm{cbd}_S\, s$ are bounded by M and let n denote the cardinality of S. Then Algorithm 6.1 runs in $O(2M^2 n)$ time when S is implemented as a bit array (bitmap) and in $O(2M^2 n \log n)$ time when S is implemented as a binary search tree. In particular, in the case of a cubical complex implemented as a bitmap Algorithm 6.1 runs in $O(2d^2 n)$, where d denotes the embedding dimension of the cubical set.*

Finally let us remark that Algorithm 6.1 may be easily extended so that its "`while`" loop is also used to separate the connected components of the input. As we explained in Sect. 5 such a separation is needed if one wants to fully benefit from coreductions in the case of simplicial and cubical complexes. The separation of components may be done as preprocessing but combining it with Algorithm 6.1 improves the overall efficiency.

7 Numerical Experiments

The coreduction homology algorithm for cubical sets and $\mathbb{Z}$ coefficients has been implemented by the first author. The implementation, in the following referred to as CR, is available from [18]. This Web page also provides documentation, the benchmark program and sample inputs used in the numerical experiments presented in this section. CR also constitutes a part of the libraries [29] and [28]. A version for simplicial complexes and general S-complexes with arbitrary ring coefficients is in preparation. The cubical version of CR accepts on input a list of elementary cubes as well as bitmaps. It switches to the implementation of [11] when no more coreductions are available. The output contains Betti numbers and torsion numbers.

We compared CR with all cubical homology programs available from the Computational Homology Project Web page [28]. These programs and the benchmark software may be easily compiled in one executable, which makes the comparison straightforward and reliable. The strongest competitors are the following two programs

BK Geometrically controlled algebraic reductions by W. Kalies [13], based on [14]
AS Reductions via acyclic subspace by M. Mrozek [18], based on [20].

In particular, both BK and AS significantly outperform the standard homology algorithm based on sparse matrix implementation of Smith Normal Form (see [20]). Therefore we restrict the presentation of the comparison only to the case of CR, BK and AS with the exception of dimension three, where we also add the results for AS3, a version of AS based on configuration lookup tables. AS3 is particularly fast, but it is not available in dimensions higher than three due to double exponential growth of the number of configurations.

Given a cubical set $X \subset \mathbb{R}^d$, put

$$\begin{aligned}
\mathcal{K}(X) &:= \left\{Q \mid Q \subset X,\ Q \text{ is an elementary cube in } \mathbb{R}^d\right\},\\
\mathcal{K}_q(X) &:= \left\{Q \in \mathcal{K}(X) \mid \dim Q = q\right\},\\
|X| &:= \operatorname{card} \mathcal{K}(X),\\
|X|_q &:= \operatorname{card} \mathcal{K}_q(X),\\
\dim X &:= \max\left\{\dim Q \mid Q \in \mathcal{K}(X)\right\},\\
\operatorname{emb} X &:= d.
\end{aligned}$$

We will refer to $\dim X$ as the *dimension* of X and to $\operatorname{emb} X$ as the *embedding dimension* of X. We say that X is a *full cubical set* if $\bigcup \mathcal{K}(X) = \bigcup \mathcal{K}_d(X)$. The cubical homology algorithms are oriented primarily on full cubical sets, because full cubical sets appear in a natural way in raster graphics, where pixels or voxels correspond to full elementary cubes. Full cubical sets may be stored very efficiently as bitmaps. General cubical sets may also be stored as bitmaps, but then the memory requirement doubles for every dimension.

Obviously, the running time of a cubical homology algorithm depends on the size of the input and the dimension of the space. In case of a general cubical set the size of input is $|X|$; however, in the case of a full cubical set there is no need to store the lower dimensional cubes, so in this case by the size of input we mean $|X|_d$. Note that

$$|X|_d \leq |X| \leq 3^d |X|_d,$$

hence in a fixed embedding dimension these two measures are asymptotically equivalent.

Every cubical set stored as a bitmap may be transformed with no cost into a full cubical set with the same homology and the same size. From the geometrical point of view this is achieved by blowing up every element of $\mathcal{K}(X)$ into a full cube. On the bitmap level this operation is just a reinterpretation of bits, so no running cost is involved. The opposite operation may be performed too, but this carries some cost depending linearly on the size of the set and exponentially on the dimension of the space.

The AS and BK implementations are designed for full cubical sets but without any extra cost, via the reinterpretation described above, they deal with general cubical sets too. In contrast, CR requires a general cubical set on input. When a full cubical set is provided, it must be first decomposed into all its faces. For this reason we study two versions of CR. The first one, denoted CRg, assumes a general cubical set on input, i.e. all the faces must already be included in the input. The second one, denoted CRf, accepts a full cubical set on input, generates all the lower dimensional faces and continues with the coreduction process. In the case of CRf the cost of generating the lower dimensional faces constitutes a part of the total cost of this algorithm. In order to compare the algorithms on full cubical sets we send general or full cubical sets to CRf, AS and BK. By using CRf we ensure that even in the case of a general cubical set the coreduction algorithm, similarly to AS and BK, first reinterprets the data as if it were a full cubical set, then generates the lower dimensional faces and only after that the proper algorithm starts. In order to compare the algorithms on general cubical sets, we send general cubical sets to CRg, AS and BK.

For every tested set we provide its dimension d and its embedding dimension e. Unless otherwise stated, the times presented in this section were obtained on a 3.6 GHz PC with Pentium 4 processor and 2 GB RAM running Windows XP with virtual memory turned off. All the software were compiled with gcc compiler version 3.4.2 ported for MS Windows XP.

Table 1 presents the tests performed on several rescalings of six general cubical sets: four cubical spheres of dimension $d = 2, 3, 4, 5$ and embedding dimension $e = 3, 4, 5, 6$, respectively, Bing's house [2] embedded in $\mathbb{R}^3$ and Klein bottle embedded in $\mathbb{R}^4$. Bing's house and Klein bottle are built of two-dimensional elementary cubes. The bottom row of every table contains a rough measure of complexity α of the programs obtained by finding the best fit of the data to the function $T = cn^{\alpha}$. Note that in theory α cannot be less than one. However, the situation when a substantial cost independent of the size of input is present may show up as α less than one, especially when and the remaining cost is close to linear.

According to the results of experiments gathered in Table 1, CR shows best performance with the exception of full cubical sets in dimension 3, when AS3 is slightly better. CR's advantage over the other programs is particularly strong in higher embedding dimensions.

The topology of examples gathered in Table 1 is relatively simple. To see how the algorithm behaves under more complicated topology, we tested it on some random cubical sets. To generate a random full cubical set X in $\mathbb{R}^d$ we run through every full elementary cube Q contained in $[0, n]^d$ for some fixed integer $n > 0$ and selected Q

Table 1 Homology computation time in seconds for various sets and various algorithms

Size	CRg	CRf	AS3	AS	BK
86402	0.05	0.38	0.14	6.7	23.3
117602	0.08	0.50	0.19	9.6	38.0
153602	0.09	0.67	0.27	12.7	87.0
194402	0.13	0.86	0.34	16.2	60.7
240002	0.14	1.08	0.41	20.8	80.1
290402	0.17	1.28	0.53	26.7	91.1
α	1.0	1.0	1.1	1.1	1.1

Cubical sphere (dim = 2, emb = 3)

(a)

Size	CRg	CRf	AS3	AS	BK
74341	0.05	0.44	0.27	12.4	32
132321	0.08	0.80	0.39	21.9	58
206901	0.13	1.27	0.61	34.6	76
298081	0.19	1.80	0.84	49.7	182
405861	0.23	2.52	1.23	71.3	221
530241	0.31	3.34	2.02	94.2	385
α	1.0	1.0	1.0	1.0	1.3

Bing's House (dim = 2, emb = 3)

(b)

Size	CRg	CRf	AS	BK
1776	<0.016	0.02	0.5	5
13920	0.016	0.11	3.6	90
46800	0.031	0.42	11.9	408
110784	0.093	1.00	27.9	1911
216240	0.156	2.02	55.3	4320
α	1.0	1.0	1.0	1.4

Cubical sphere (dim = 3, emb = 4)

(c)

Size	CRg	CRf	AS	BK
1382	<0.015	0.03	0.4	3
12522	0.015	0.14	3.2	32
34830	0.047	0.42	8.4	94
68306	0.079	0.95	16.4	201
112950	0.141	1.92	27.6	335
α	1.0	1.0	1.0	1.1

Klein bottle (dim = 2, emb = 4)

(d)

Size	CRg	CRf	AS	BK
1298	<0.015	0.016	4	25
4442	<0.015	0.078	16	129
9458	0.016	0.172	43	358
16346	0.031	0.297	87	649
25106	0.047	0.469	146	1089
α	–	1.1	1.3	1.3

Cubical sphere (dim = 4, emb = 5)

(e)

Size	CRg	CRf	AS	BK
728	<0.015	0.05	52	594
3944	<0.015	0.17	433	–
9720	<0.015	0.44	1311	–
18056	0.031	0.89	2590	–
28952	0.063	1.48	4344	–
α	–	0.9	1.2	–

Cubical sphere (dim = 5, emb = 6)

(f)

to X with a fixed probability $p \in (0, 1)$. To generate a random general cubical set X in $\mathbb{R}^d$ we first run through every zero-dimensional cube (vertex) Q contained in $[0, n]^d$ for some fixed integer $n > 0$ and we selected Q to the set of vertices of X with a fixed frequency $p \in (0, 1)$. Then we added to X every elementary cube whose all vertices belonged to the generated set of vertices. The distribution of Betti numbers of such random cubical sets crucially depends on the probability p. One can expect that when p is not large enough, the chances of closing a cycle are too small. On the other hand, when p is too large the chances for the cycle to be nontrivial are also small, so that

Table 2 Homology computation time in seconds for random cubical sets in $\mathbb{R}^3$ and their rescalings. The sets were generated and processed as full cubical sets

Size	CR	AS3	AS	BK
1592318	1.03	1.25	198	2716
2974304	1.81	2.11	379	3549
4984602	2.92	3.38	646	7048
7740620	4.41	4.83	1012	–
α	0.9	0.9	1.0	–
$p = 0.3$, dim = emb = 3				
Betti: 5, 676, 5				

Size	CR	AS3	AS	BK
2432700	2.09	3.2	323	4481
4599206	3.64	5.2	622	–
7771072	5.75	7.6	1243	–
12139338	8.56	10.9	2168	–
α	0.9	0.8	1.2	–
$p = 0.5$, dim = emb = 3				
Betti: 1, 563, 38				

Size	CR	AS3	AS	BK
3196734	4.7	12.5	659	5900
6113878	8.0	19.4	1274	–
10412662	14.7	27.5	2128	–
16360302	18.8	37.6	2878	–
α	0.9	0.7	0.9	–
$p = 0.7$, dim = emb = 3				
Betti: 1, 60, 297				

Size	CR	AS3	AS	BK
3874985	4.1	12.3	697	7467
7497607	7.3	20.9	1460	–
12872733	11.8	29.7	2546	–
20345819	18.0	43.3	4002	–
α	0.9	0.7	1.1	–
$p = 0.9$, dim = emb = 3				
Betti: 1, 1, 417				

the maximum number of nontrivial cycles is observed for some intermediate value of p that depends on q. However, more numerical experiments are needed in order to get a better understanding of this dependence and to formulate some hypotheses.

Table 2 shows homology computation times for rescalings of four cubical sets in $\mathbb{R}^3$ generated as full cubical sets with the probability p of selecting a full cube to the set, respectively, 0.3, 0.5, 0.7 and 0.9. Table 3 shows the time of homology computations of a few subsets of a random general cubical set generated in dimension 4 and 5. The performance of CRg for these examples is skipped. However, let us mention that in these cases CRg is 6 to 12 times better than CRf. Also here the superiority of CR is clearly visible.

To see how the performance of CR depends on the embedding dimension, Table 4 presents computation times of CRf and AS for random cubical sets whose vertices were selected with fixed frequency 0.3 and whose size is of the same order of magnitude but the embedding dimension varies from 3 to 7. From the last column of this table we see that the superiority of CR over AS grows with dimension.

Table 5 presents four extreme cases of homology computations by CR. The first two cases are taken from a numerical simulation of Cahn–Hillard equations [10]. The other two cases concern the minimal face consistent cubical set containing all the solutions on some energy sublevel sets for a randomly chosen k-SAT problem with 12 variables and 50 clauses, i.e. close to the critical value 4.3 of the ratio of the number of clauses to the number of variables, where the high complexity of the k-Sat problem is conjectured to concentrate [15].

Table 3 Homology computation time in seconds for random cubical sets in $\mathbb{R}^3$ and $\mathbb{R}^4$ and their subsets. The sets were generated as random general cubical sets and converted to full cubical sets before processing

Size	Betti	CRf	AS	BK	Size	Betti	CRf	AS	BK
10348	541, 279	0.125	0.72	58	62636	24, 2988, 20	0.53	20	598
13434	650, 406	0.156	1.05	188	116284	36, 5552, 38	1.06	52	2430
16470	769, 517	0.187	1.27	280	168750	48, 8060, 54	1.56	102	5490
19363	898, 625	0.218	1.52	383	222480	54, 10493, 71	2.09	160	13489
α		0.9	1.2	3.0	α		1.1	1.6	2.4
$p = 0.3$, dim = emb = 4					$p = 0.6$, dim = emb = 4				

Size	Betti	CRf	AS	BK	Size	Betti	CRf	AS	BK
14883	31, 1044, 2	0.41	14	3014	34045	1, 0, 8, 14	0.44	51442	5148
32674	40, 2256, 6	0.86	38	13619	56211	1, 0, 8, 30	0.74	–	–
50549	49, 3419, 13	1.36	73	–	78812	1, 0, 10, 51	1.09	–	–
68102	53, 4627, 20	1.88	113	–	100558	1, 0, 12, 67	1.48	–	–
α		1.0	1.4	–	α		1.1	–	–
$p = 0.5$, dim = emb = 5					$p = 0.95$, dim = emb = 5				

Table 4 Dependence of the computation time on dimension

Dimension	Size	Betti	CRf	AS	AS/CRf
4	13730	610, 402	0.157	0.97	6.2
5	12135	330, 685	0.469	4.8	10.2
6	12162	217, 987	1.48	22.4	15.1
7	14031	143, 1406	7.23	188	26.0

Table 5 Computation time in the case of large size and large dimension

Dimension	Size	Betti	CRg	CRf
3	8392997	1, 1057	–	4.38
4	350000000	1, 7, 352	–	935
12	135351	1, 0, 0, 1, 4, 5, 0	0.77	–
12	332309	1, 0, 0, 0, 0, 1, 1	1.92	–

Direct comparison of CR with homology programs not available in [28] is more complicated, because of difficulties in compiling various packets together and differences in input formats. However, some rough comparison is possible. Recently A. Urbańska [25] compared CRf with homology algorithms using sparse implementations of Smith diagonalization algorithms available in the LinBox library [30]. The experiments were performed on SGI Altix with 64 Itanium 2 processors. She found that the algorithms in LinBox cannot compete with CR, at least not for the class of problems, where the complex is large and the homology to be computed is simple. For instance, the computation of the homology of Klein bottle built out of 380 full four-dimensional cubes with CR turned out to be more than 20 times faster than with LinBox. And in the case of a cubical set built out of about 8 million three-dimensional cubes CR was more than 1300 times faster.

Zomorodian and Carlsson presented in [27] the times resulting from the 2.2 GHz Pentium processor homology computations of a simplicial representation of Klein bottle consisting of 12,000 simplices. The times are 0.01 s for $\mathbb{Z}_2$ coefficients, 0.23 s for $\mathbb{Z}_3$, $\mathbb{Z}_5$ and $\mathbb{Z}_{3203}$ coefficients and 0.5 s for rational coefficients. The most similar case in our experiments is Klein bottle treated as a general cubical set built out of 12,522 elementary cubes of dimension at most 2. Table 1d shows 0.015 s as the computation time for this case. When rescaled from the 3.6 GHz clock to the 2.2 GHz clock this would be 0.025 s. This is 2.5 times slower than the case of Zomorodian and Carlsson for $\mathbb{Z}_2$ coefficients but 9 to 20 times faster than the case of field coefficients. Let us recall that CR computes the homology over $\mathbb{Z}$ coefficients.

8 Complexity

In this section we discuss the computational complexity of the coreduction homology algorithm. Obviously it depends on the size of the S-complex left after the reduction. We do not yet have a deep understanding of the structure of the output of Algorithm 6.1. For this, more numerical experiments are needed. We do have a conjecture and a partial result, which to some extent explains why we believe the conjecture might be true. This partial result is also important in applications to some algorithms in rigorous numerics of dynamical systems (see [19]).

In the following we restrict our attention to S-complexes coming from cubical sets and we also assume that the queue used to keep candidates for coreductions in Algorithm 6.1 is sorted with respect to the lexicographic order induced from $\mathbb{R}^d$. Note that the complexity of such a modified algorithm is $O(n\log n)$ regardless of the storage method used for sets. The first restriction is purely for simplicity. What concerns the other restriction is irrelevant for the complexity result discussed in this section, because the argument presented does not have any chance to go below $O(n\log n)$ anyway.

Given a cubical set X put

$$\gamma(X) := \frac{|X'|}{|X|},$$

where X' denotes the S-complex returned by Algorithm 6.1. The following proposition is straightforward.

Proposition 8.1 *Assume that for a certain class of cubical sets we have*

$$\gamma(X) = O\left(\frac{1}{|X|^{\mu}}\right) \tag{5}$$

for some $\mu \in (0,1)$. Then the complexity of finding homology in this class by means of Algorithm 6.1 followed by another homology algorithm applied to X' is

$$O\big(|X|^{\alpha(1-\mu)}\big)$$

if the complexity of the other homology algorithm is $O(|X|^\alpha)$ and $\mu < 1 - \frac{1}{\alpha}$.

The question is: Under what assumptions can we guarantee that (5) is satisfied? We have the following conjecture.

Conjecture 8.2 *Consider a class of q-dimensional cubical sets such that any two of them are homeomorphic under a certain class of homeomorphisms. Then γ restricted to this class satisfies (5) with $\mu = \frac{1}{q}$.*

If the conjecture is true, then in particular the coreduction algorithm reduces the complexity of computing homology in a fixed topology class of a two-dimensional set from $O(|X|^\alpha)$ to $O(|X|^{\frac{\alpha}{2}})$. Recall that by the dimension of a cubical set we mean the maximal dimension of the elementary cubes contained in the cubical set. For instance, the Klein bottle may be represented as a two-dimensional cubical set although the minimum embedding dimension for this set is four. Also note that the reduction algorithm reduces the dimension of the set embedded in $\mathbb{R}^d$ to $d-1$.

In applications to rigorous numerics of dynamical systems certain algorithms are used (see [19]) that produce cubical sets whose homology must be verified. To guarantee certain properties, these algorithms frequently subdivide the cubical sets, so that although the topology of the outcome is relatively simple, the sets are built of huge chunks of frequently subdivided cubes. In order to stick to cubical sets whose vertices have integer coefficients as considered in this paper, it is convenient to replace the process of subdivision by the process of rescaling. By a rescaling we mean a homeomorphism of the form

$$R_n : \mathbb{R}^d \ni x \to nx \in \mathbb{R}^d$$

for some integer $n > 0$. By an n-rescaling of a cubical set X we mean the image of X under R_n.

We have the following theorem

Theorem 8.3 *Consider the class of all n-rescalings of a fixed q-dimensional cubical set X_0. Then γ restricted to this class satisfies (5) with $\mu = \frac{1}{q}$. In particular, the complexity of finding homology in this class is*

$$O\big(|X|^{\alpha(1-\frac{1}{q})}\big).$$

Before we sketch the proof of this theorem, let us consider the outcome of Algorithm 6.1 on the rescalings of cubical Bing's house by factors of 2 and 5, respectively (presented in Fig. 3). Note that the reduced sets contain no vertices and every edge left has exactly two elements in its coboundary except one edge whose coboundary cardinality is three. Moreover, the reduced set expands under the rescaling only linearly. This is a hint how to proceed with the proof.

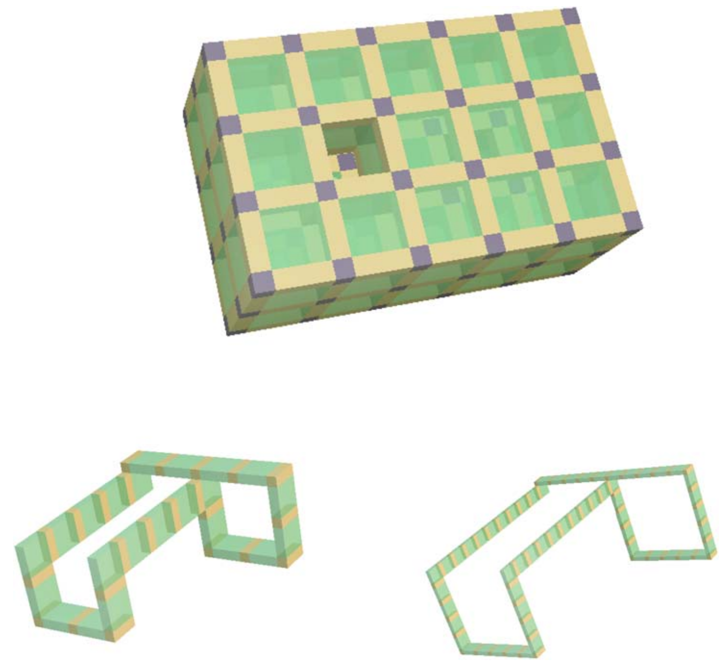

Fig. 3 Cubical Bing's house (*top*) and the outcome of Algorithm 6.1 on two Bing's house rescalings: by factor 2 (*bottom left*) and by factor 5 (*bottom right*)

Proof We sketch the proof for $q = 2$. The details, especially for $q > 2$, are technically complicated and will be published elsewhere. Since the considered dimension is 2, we use the terminology vertices, edges and faces, respectively, for zero-, one- and two-dimensional elementary cubes.

Let X be a cubical set in the considered class and let $n > 0$ be an integer satisfying $X = R_n(X_0)$. Denote by X' the outcome of Algorithm 6.1 applied to X. Obviously $|X|_2 = n^2|X_0|_2$. It is enough to show that

$$|X'|_2 \le 4n|X_0|_2. \tag{6}$$

Indeed, if (6) is satisfied, then

$$|X'| \le 9|X'|_2 \le 36n|X_0|_2 = 36|X_0|_2^{\frac{1}{2}}|X|^{\frac{1}{2}} = O\big(|X|^{\frac{1}{2}}\big)$$

and consequently $\gamma(X) = O(\frac{1}{|X|^{\frac{1}{2}}})$.

It remains to prove (6). First, observe that we can assume that X is connected, because if not, we can apply the algorithm and the argument separately to each connected component of X. Let us assume that all vertices, edges and faces in X are ordered under the lexicographic order induced from $\mathbb{R}^d$.

Given a face F_0 in X_0, the minimal edge in the boundary of F_0 in X_0 is called the *base* of F_0. An edge is called a *support* in X_0 if it is a base of a face in X_0. The *index* of a support E_0 in X_0 is the number of all faces in X_0 which are under E_0 in the lexicographic order. We say that an edge E in X is a *skeleton* covered by an edge E_0 in X_0, if it is contained in the image of E_0 under R_n. We say that a vertex, an edge or a face in X is *internal* if it does not contain the image of any vertex in X_0 under R_n.

First we claim that Algorithm 6.1 applied to X removes all internal skeleton edges of X covered by a support of X_0. Note that any edge removed by Algorithm 6.1 is removed either as a member of a vertex-edge or edge-face coreduction pair. Moreover, every vertex is removed, because the connectedness of X implies that X' contains no vertices. A case study argument based on tracing the reduction of vertices, the location of the edge with respect to the other edges and the order in which the edges are removed from the queue shows that all the skeleton internal edges covered by a support of zero index are removed in a vertex-edge reduction. A similar induction argument shows that all the internal skeleton edges covered by a support of positive index as well as all the internal faces are removed in an edge-face reduction. Therefore the number of internal faces in X' which are covered by one face in X_0 does not exceed $4n$, which proves (6). □

We believe that in principle a similar proof should go through for the class of homeomorphsims which factorize as a composition of rescalings and other homeomorphisms which keep the size bounded.

9 Conclusions and Final Remarks

The method of coreductions introduced in this paper is a novel way of reducing large complexes in linear time to proportions in which the classical cubical algorithms can take over. Experiments based on cubical sets show that coreduction homology algorithm performs much better than other cubical homology algorithms available so far, particularly for low-dimensional sets in high dimensions. Moreover it is possible to adapt the coreduction homology algorithm to maps. The first tests show that in the case of maps the gain over the existing homology algorithms for maps is even stronger. Details will be published elsewhere.

Acknowledgements The authors thank an anonymous referee for some valuable comments which helped to improve and correct the presentation of the paper.

References

1. Basu, S.: On bounding the Betti numbers and computing the Euler characteristic of semi-algebraic sets. Discrete Comput. Geom. **22**, 1–18 (1999)
2. Bing, R.H.: Some aspects of the topology of 3-manifolds related to the Poincaré Conjecture. In: Saaty, T.L. (ed.) Lectures on Modern Mathematics, vol. II, pp. 93–128. Wiley, New York (1964)
3. Delfinado, C.J.A., Edelsbrunner, H.: An incremental algorithm for Betti numbers of simplicial complexes on the 3-sphere. Comput. Aided Geom. Des. **12**, 771–784 (1995)
4. Donald, B.R., Chang, D.R.: On the complexity of computing the homology type of a triangulation. In: Proc. 32nd Ann. IEEE Sympos. Found. Comput. Sci., pp. 650–661 (1991)
5. Duff, I., Erisman, A., Reid, J.: Direct Methods for Sparse Matrices. Oxford University Press, London (1986)

6. Dumas, J.-G., Saunders, B.D., Villard, G.: On efficient sparse integer matrix Smith normal form computations. J. Symb. Comput. **32**, 71–99 (2001)
7. Dumas, J.-G., Heckenbach, F., Saunders, D., Velker, V.: Computing simplicial homology based on efficient smith normal form algorithms. In: Algebra, Geometry and Software Systems, pp. 177–207. Springer, Berlin (2003)
8. Edelsbrunner, H., Letscher, D., Zomorodian, A.: Topological persistence and simplification. Discrete Comput. Geom. **28**, 511–533 (2002)
9. Friedman, J.: Computing Betti numbers via combinatorial Laplacians. In: Proc. 28th Ann. ACM Sympos. Theory Comput., pp. 386–391 (1996)
10. Gameiro, M., Mischaikow, K., Wanner, Th.: Evolution of pattern complexity in the Cahn–Hilliard theory of phase separation. Acta Mater. **53**, 693–704 (2005)
11. Kaczynski, T., Mrozek, M., Ślusarek, M.: Homology computation by reduction of chain complexes. Comput. Math. Appl. **35**, 59–70 (1998)
12. Kaczynski, T., Mischaikow, K., Mrozek, M.: Computational Homology. Applied Mathematical Sciences, vol. 157. Springer, New York (2004)
13. Kalies, W.: Chom—A Cubical Homology Program (1999). http://www.math.fau.edu/kalies/chom.html
14. Kalies, W., Mischaikow, K., Watson, G.: Cubical approximation and computation of homology. In: Conley Index Theory, vol. 47, pp. 115–131. Banach Center Publications, Warsaw (1999)
15. Kirkpatrick, S., Selman, B.: Critical behaviour in the satisfiability of random Boolean expressions. Science **264**, 1297 (1994)
16. Massey, W.S.: Homology and Cohomology Theory. Dekker, New York (1978)
17. Mischaikow, K., Mrozek, M., Pilarczyk, P.: Graph approach to the computation of the homology of continuous maps. Found. Comput. Math. **5**, 199–229 (2005)
18. Mrozek, M.: Homology Software (2006). http://www.ii.uj.edu.pl/~mrozek/software/homology.html
19. Mrozek, M.: Index pairs algorithms. Found. Comput. Math. **6**, 457–493 (2006)
20. Mrozek, M., Pilarczyk, P., Zelazna, N.: Homology algorithm based on acyclic subspace. Comput. Math. Appl. (2008, accepted)
21. Mrozek, M., Batko, B.: Homology of representable sets (2008, submitted)
22. Munkres, J.R.: Elements of Algebraic Topology. Addison-Wesley, Reading (1984)
23. Steenrod, N.E.: Regular cycles of compact metric spaces. Ann. Math. **41**, 833–851 (1940)
24. Storjohann, A.: Near optimal algorithms for computing Smith normal form of integer matrices. In: Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation, ISAAC 1996, pp. 267–274 (1996)
25. Urbańska, A.: Smith normal form algorithms for sparse matrices with applications to homology computations. M. Sc. Thesis, Jagiellonian University, Kraków (2007) (in Polish, with English Abstract)
26. Yannakakis, M.: Computing the minimum fill-in is NP-complete. SIAM J. Matrix Anal. Appl. **2**, 77–79 (1981)
27. Zomorodian, A., Carlsson, G.: Computing persistent homology. Discrete Comput. Geom. **33**, 249–274 (2005)
28. Computational Homology Project: http://chomp.rutgers.edu
29. Computer Assisted Proofs in Dynamics: http://capd.wsb-nlu.edu.pl
30. Project LinBox: Exact computational linear algebra: http://www.linalg.org