Simplicial complexes in Sage

J. H. Palmieri

University of Washington

Seattle, 17 May 2009

A simplicial complex is defined by specifying a set V of (vertices) and a set $\mathcal{F}$ of subsets of V (simplices) closed under taking subsets: if S is a simplex, then so is every subset of S.

Let $V = \{0, 1, 2, \ldots, n\}$.
$S = \{i\} \leftrightarrow$ vertex i
$S = \{i, j\} \leftrightarrow$ edge between i and j
$S = \{i, j, k\} \leftrightarrow$ triangle determined by i, j, k
etc.

An efficient way to define one: specify V and the simplices which are maximal w.r.t. inclusion.

Let K be a simplicial complex. The homology of K, written $H_*(K)$, is defined as follows: form a chain complex $C_*(K)$ with $C_n(K)$ equal to the free abelian group on the n-dimensional simplices of K, and with differential $d_n : C_n(K) \to C_{n-1}(K)$ defined by

$$d_n([v_0, v_1, \ldots, v_n]) = \sum_{i=0}^{n} (-1)^i [v_0, \ldots, \hat{v}_i, \ldots, v_n],$$

where $[v_0, \ldots, v_n]$ is the simplex determined by the listed vertices and the hat $\hat{v}_i$ means to omit that vertex.

Can check: $d_n \circ d_{n+1} = 0$.

Define: $H_n(K) = \dfrac{\ker d_n : C_n(K) \to C_{n-1}(K)}{\operatorname{im} d_{n+1} : C_{n+1}(K) \to C_n(K)}$.

You can also work with coefficients: for any commutative ring R, define

$$H_n(K;R) = \frac{\ker d_n : C_n(K) \otimes R \to C_{n-1}(K) \otimes R}{\operatorname{im} d_{n+1} : C_{n+1}(K) \otimes R \to C_n(K) \otimes R}.$$

Also have relative homology: if L is a subcomplex of K, then define $C_*(K,L) = C_*(K)/C_*(L)$, and define

$$H_n(K,L) = \frac{\ker d_n : C_n(K,L) \to C_{n-1}(K,L)}{\operatorname{im} d_{n+1} : C_{n+1}(K,L) \to C_n(K,L)}.$$

(Note that $C_*(K,L)$ is free abelian on the simplices which are in K but not in L.)

Implemented in Sage:

- Chain complexes
- Simplicial complexes
  - define by specifying vertices and facets, or
  - choose from a list of pre-defined complexes
- Various operations on simplicial complexes: join, product, barycentric subdivision, suspension, cone, Stanley-Reisner ring, . . .

I'm mainly interested in computing homology. The main computational issue is:

- computing the Smith normal form of the matrix for the differential d_n.

(Literature review: [DHSW], . . .)

Some timings:

```
sage: time S62 = simplicial_complexes.\
  NotIConnectedGraphs(6,2)
CPU times: user 0.06 s, sys: 0.01 s, total: 0.07 s
Wall time: 0.91 s
sage: time C62 = S62.chain_complex()
CPU times: user 8.37 s, sys: 0.47 s, total: 8.84 s
Wall time: 20.66 s
```

```
sage: S62.f_vector()
[1, 15, 105, 455, 1365, 3003, 4945, 5715, 3990,
   1470, 306, 30]
sage: sum(S62.f_vector())
21400
sage: C62.differential()
{0: [],
 1: 15 x 105 sparse matrix over Integer Ring,
 2: 105 x 455 sparse matrix over Integer Ring,
 3: 455 x 1365 sparse matrix over Integer Ring,
 4: 1365 x 3003 sparse matrix over Integer Ring,
 5: 3003 x 4945 sparse matrix over Integer Ring,
 6: 4945 x 5715 sparse matrix over Integer Ring,
 7: 5715 x 3990 sparse matrix over Integer Ring,
 8: 3990 x 1470 sparse matrix over Integer Ring,
 9: 1470 x 306 sparse matrix over Integer Ring,
```

```
sage: time C62.homology(base_ring=GF(2))
CPU times: user 3.50 s, sys: 0.20 s, total: 3.70 s
Wall time: 3.91 s
```

On the other hand,

```
sage: time C62.homology()
```

takes hours on my iMac. For example:

```
sage: mat = C62.differential(5); mat
3003 x 4945 sparse matrix over Integer Ring
sage: time mat.elementary_divisors()
CPU times: user 1926.47 s, sys: 71.45 s, total: 199
Wall time: 2033.12 s
```

One way to repair this: use smaller matrices.

Observation: if L is a subcomplex of K with $H_*(L) = 0$, then $H_*(K, L) \cong H_*(K)$.

So look for a large subcomplex L with trivial homology. Then the matrices involved in computing $H_*(K, L)$ will be smaller than those used to compute $H_*(K)$:

```
sage: S62 = simplicial_complexes.NotIConnectedGraph
sage: time L62 = S62._contractible_subcomplex()
CPU times: user 17.84 s, sys: 0.03 s, total: 17.87
Wall time: 17.97 s
sage: time C62 = S62.chain_complex(subcomplex=L62)
CPU times: user 1.91 s, sys: 0.01 s, total: 1.93 s
Wall time: 1.99 s
```

```
sage: S62.f_vector()
[1, 15, 105, 455, 1365, 3003, 4945, 5715, 3990,
   1470, 306, 30]
sage: [a-b for (a,b) in zip(S62.f_vector(), \
   L62.f_vector())]
[0, 0, 0, 0, 0, 0, 24, 158, 236, 96, 20, 2]

sage: time C62.homology()
CPU times: user 0.18 s, sys: 0.01 s, total: 0.18 s
Wall time: 0.21 s
{0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: Z^24,
   8: 0, 9: 0, 10: 0}
```

In summary:

```
sage: S62 = simplicial_complexes.\
   NotIConnectedGraphs(6,2)
sage: time S62.homology()
CPU times: user 20.49 s, sys: 0.18 s, total: 20.67
Wall time: 21.74 s
{0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: Z^24,
   8: 0, 9: 0, 10: 0}
```

Information is also cached:

```
sage: time S62.homology()
CPU times: user 0.21 s, sys: 0.01 s, total: 0.22 s
Wall time: 0.23 s
{0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: Z^24,
   8: 0, 9: 0, 10: 0}
```

Comparison to other implementations:

- **Mathematica**
- **Maple** Moise: “Moise is not designed to be an optimal way to do these calculations”
- **Gap** : has an optional package which is much faster than Sage (10–100 times faster?)
  Package is written by Dumas, Heckenbach, Saunders, Welker
  I had a hard time installing it. . .

Ways to speed up our version:

- Rewrite parts of the code in Cython
- Speed up Smith normal form computations – sparse implementation? Also see [DHSW]
- Reduce mod p? Open problem. . .

Other things to consider implementing:

- simplicial sets – See Kenzo
- cubical sets – see Chomp (?)
- delta complexes, almost-simplicial complexes
- CW complexes