BioGeometry NEWS
February 2005
biogeometry.cs.duke.edu

Research **Persistence Algorithm Takes Cubic Time in Worst Case** by Dmitriy Morozov

Given a sequence of N simplices, we consider the sequence of sets K_i consisting of the first i simplices, for $1 \le i \le N$. We call the sequence of K_i a *filtration* if all the K_i are simplicial complexes. In this note, we describe a filtration of a simplicial complex of N simplices on which the algorithm Pair-Simplices of Edelsbrunner, Letscher and Zomorodian [1] performs $\Omega(N^3)$ operations. The existence of this filtration should be contrasted to the experimentally observed only slightly superlinear running time for filtrations that arise from applications.

We describe the space as well as the ordering on the simplices. Let $n = \lfloor (N+29)/7 \rfloor$, $v = \lfloor (n-1)/2 \rfloor$, and note that both n and v are in $\Omega(N)$. In our filtration, all vertices appear before all edges, and all edges appear before all triangles. We index the vertices separately from the edges separately from the triangles. Some edges will receive a negative index, which is done for simplicity to indicate that they appear before the edges with positive labels (see Figure 2).

Figure 1 illustrates the construction of our space as well as the assignment of indices to the simplices. Starting with triangle ABC, we add v vertices inside the triangle in the following manner: we place the first vertex V_1 near the middle of edge AB, the second vertex V_2 near the middle of AV_1, V_3 near the middle of BV_1, V_4 near AV_2, V_5 near BV_3, V_6 near V_1V_2, V_7 near V_1V_3, and so on, moving from both ends inwards at each stage. The edges joining C with the V_i are the first to appear in the filtration, each one merging the component containing C with V_i. These edges are not important in our argument, so we do not label them. Edge AB gets index 1, and the remaining edges are assigned indices from the ends inwards similar to the vertices: AV_1 gets n, BV_1 gets $n-1$, AV_2 gets $n-2$, BV_3 gets $n-3$, V_1V_2 gets $n-4$, V_1V_3 gets $n-5$, and so on (see Figure 1). Similarly, the triangles are assigned indices from the ends inwards in stages: ABV_1 gets 1, AV_1V_2 gets 2, BV_1V_3 gets 3, AV_2V_4 gets 4, BV_3V_5 gets 5, and so on. We call these triangles the *base triangles*.

In addition, we place $n-1$ vertices above the plane of triangle ABC, one above each edge AV_i, BV_j, and V_iV_j, and join them to the vertices of those edges (Figure 1 depicts only two of the $n-1$ such vertices). One of the edges joining the vertex above the edge k to its endpoints will merge the component containing the endpoint of the edge k with the vertex above the plane. We do not label this edge. The other edge gets index $k-(n+1)$, which is negative. The triangle formed gets an index larger than v, so that the triangles not in the plane of ABC appear last in the filtration. We call them *fin triangles*.

Consider what happens when algorithm Pair-Simplices given in [1] processes the described filtration. There are two interesting parts to its execution. First, when the base triangles 1 to v are processed, the edges n to $n-v$ build up lists of $\Omega(n)$ simplices each (see Figure 3). Second, when the fin triangles $v+1$ to $v+n$ are processed, the search for the corresponding edges goes through all the lists shared at these edges, merging lists

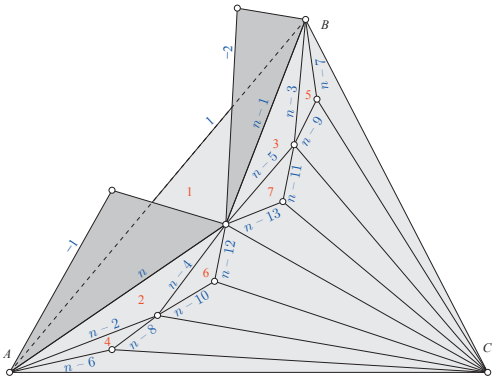


Figure 1. The underlying space of the filtration. The edge indices are blue and the triangle indices are red. Each edge with label larger than 1 is the base of a triangle coming out of the plane of the triangle ABC (only triangles with base edges labelled n and $n-1$ are shown).

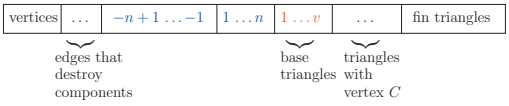


Figure 2. Simplex ordering.

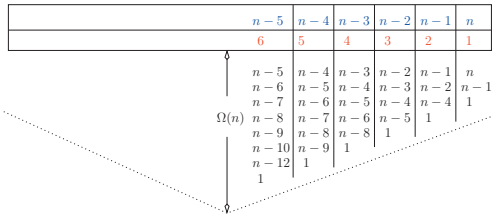


Figure 3. Processing the filtration: pairing of edges 1 through n with the base triangles.

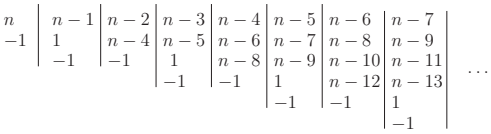


Figure 4. Processing the filtration: intermediate lists during the search for the pairing of the fin triangle above edge n.

of length $\Omega(n)$. As a result, for $\Omega(n)$ triangles we perform $\Omega(n)$ merges each of which takes time $\Omega(n)$. It follows that the total running time is $\Omega(N^3)$.

To understand this argument, let us consider what happens when we process the triangles one by one (see Figure 3). First, the base triangles are processed. Triangle 1 is paired with edge n, depositing its list of boundary edges (n, $n-1$, 1) with edge n. Triangle 2 with boundary edges (n, $n-2$, $n-4$) causes a collision at edge n, the two lists merge and the result ($n-1$, $n-2$, $n-4$, 1) gets deposited with edge $n-1$. Triangle 3 with boundary edges ($n-1$, $n-3$, $n-5$) causes a collision at edge $n-1$, the lists merge and the result ($n-2$, $n-3$, $n-4$, $n-5$, 1) is deposited with edge $n-2$. Triangle 4 with boundary edges ($n-2$, $n-6$, $n-8$) causes a collision at edge $n-2$, the lists merge and the result ($n-3$, $n-4$, $n-5$, $n-6$, $n-8$, 1) is deposited with edge $n-3$. Continuing to the last base triangle builds up lists stored with edges 1 to n and $\Omega(n)$ of these lists have length $\Omega(n)$.

Second, the fin triangles are processed. The triangle above

edge n has the labeled edges n and -1 in its boundary. It eventually gets paired with edge -1, but before that happens, the search goes through each one of the edges n through 1. To see this, note that the boundary edges cause a collision at edge n, the lists are merged to get ($n-1$, 1, -1), the new list causes a collision at edge $n-1$, and after merging we get ($n-2$, $n-4$, -1). The next collision is at edge $n-2$, after merging the list becomes ($n-3$, $n-5$, 1, -1), the collision at edge $n-3$ produces the list ($n-4$, $n-6$, $n-8$, -1), the collision at $n-4$ gives ($n-5$, $n-7$, $n-9$, 1, -1), and so on; see Figure 4. This process is repeated for each fin triangle: a similar merge pattern occurs, the only difference being that it starts at the base edge of the processed fin triangle.

Observe that some edges get cancelled when the lists are merged, but get reintroduced two merges later. The length of each intermediate list grows by one every two merges, and its length reaches $\Omega(n)$. This implies that the running time of the algorithm is cubic as claimed earlier: for each one of the $\Omega(n)$ fin triangles, $\Omega(n)$ lists of length $\Omega(n)$ are merged.

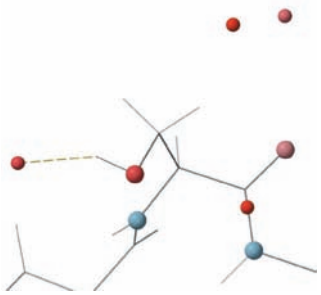
Reference:
[1] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom.* **28** (2002), 511-533.

Non-Pairwise Decomposable Solvation Models for Dynamic Programming by Andrew Leaver-Fay

There are many terms to the energy functions used to model atomic-level interactions for protein simulation and design. Most include terms for electrostatic interaction between charges, van der Waals interactions between nearly atoms, hydrogen bonding which stabilizes molecular interactions, and implicit solvation to avoid the expensive task of explicitly modeling water. Biochemists often approximate these interactions by functions of pairs of atoms, because pairwise interactions can be efficiently tabulated and looked up to avoid recalculation.

Considering only pairwise interactions may ignore important interactions in the neighborhood of an atom. For example, consider the solvation state of a ligand that may bind to a protein. In the ligand's unbound state, each hydrogen-bonding group (defined as a polar atom with its attached hydrogen atoms, if any) forms a hydrogen bond with water. In order for the protein to bind the target ligand, it must overcome the energy barrier in removing water from around the ligand. To enter the binding pocket, the ligand must break its bonds with water, incurring an energetic "desolvation penalty." Desolvation penalties are also important for the stability of a designed protein, which depends on the relative free energy of the unfolded and folded states. A serine that remains on the surface of a protein in its folded state is energetically neutral; serine that is buried is desolvated and destabilizing.

Although solvation is important for protein design, it is difficult to model. Lazaridis and Karplus (1999) introduced a pairwise-decomposable implicit solvation model which is used by several molecular modeling programs. In this model, the approach of any atom towards a polar atom is penalized proportionally to the volume of water it displaces. The approach of any atom towards a non-polar atom is rewarded in a similar proportion. Since this simple model is pairwise decomposable, it easily incorporated in search algorithms. The most obvious problem with this model is that it can over-penalize the desolvation of a polar atom, especially near the molecular surface.


A surface serine, fully solvated.

Our work on dynamic programming for finding minimum energy conformations allows us to consider non-pairwise models, so we can cap the desolvation penalty for any single hydrogen-bonding group. Once a hydrogen-bonding group has become completely desolvated, bringing additional atoms incurs no further penalty. Here the interaction energy between two atoms depends upon the other interactions these atoms have with their surroundings.

Having two solvation states in our model allow an additional refinement. We may make hydrogen-bond strength depend on the solvation state of the hydrogen-bonding group. Ariel Fernandez (2004) has found that hydrogen bonds that are shielded from water (or "wrapped") by hydrophobic groups are stronger than solvent-accessible hydrogen bonds by up to an order-of-magnitude. We will parameterize our model so that hydrogen bonds are stronger when the hydrogen-bonding groups are in their desolvated states. Consequently, hydrogen-bonding groups will seek to ease their desolvation penalty by forming good hydrogen bonds.

Within receptor design, our new model considers the solvation states of background-hydrogen-bonding groups during the side chain optimization. We consider a particular orientation of a target ligand within the active site to define a single problem instance. Previously, the ligand was assumed to be "background." In our new model, however, the optimal side chain placement depends on the solvation states of the ligand's hydrogen-bonding groups. These hydrogen-bonding groups will repel the protein's side chains unless the groups are in their desolvated states. To negate the repulsion, the optimal side chain placement will have to satisfy the ligand's hydrogen-bonding groups. Ligand hydrogen bond satisfaction is usually a post-search filter in receptor design; we hope that our non-pairwise-decomposable-energy model will allow better receptor designs by incorporating it in the search process.

We have implemented our new model within Rosetta and are testing it with simulated annealing. We are now entering the parameterization phase where we will test our model at the sequence recovery task.

The BioGeometry project is funded by the National Science Foundation under grant CCR-00-86013. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.