Department of Scientific Computing
Florida State University

Computational Expo 2011

Parallel Algorithm for Spherical Delaunay Triangulations and Spherical Centroidal Voronoi Tessellations

Doug Jacobsen
Max Gunzburger[a], Todd Ringler[b], John Burkardt[a],
Janet Peterson[a]
[a] Department of Scientific Computing, Florida State University, Tallahassee FL 32306
[b] Los Alamos National Laboratory, Los Alamos, New Mexico, 87545

Abstract Spherical centroidal Voronoi tessellations (SCVT) are used in many applications in a variety of fields, one being climate modeling. They are a natural choice for spatial discretizations on the Earth, or any spherical surface. The climate modeling community, which has started to make use of SCVTs, is beginning to focus on exa-scale computing for large scale climate simulations. Due to this, a need is brought to light for fast and efficient grid generators. Current high resolution simulations on the earth call for a spatial resolution of about 11.1km. In terms of a SCVT this corresponds to a quasi-uniform SCVT with roughly 2 million Voronoi cells. Computing this grid in serial is very expensive, and can take on the order of weeks to converge sufficiently for the needs of climate modelers. Utilizing conformal mapping techniques, as well as planar triangulation algorithms, and basic domain decomposition, this paper outlines a new algorithm that can be used to compute SCVTs in parallel, thus reducing the overall time to convergence. This reduces the actual time needed to create a grid on the Earth, as well as allows for new techniques to be explored when modeling the climate.

Goal of Research

We are attempting to improve the overall performance of spherical centroial Voronoi tessellation (SCVT) generators, by speeding up the computation of spherical Delaunay triangulations. The computation of the spherical Delaunay triangulation is the most computationally demanding part of Lloyd's algorithm, a well know algorithm for computing SCVTs. There are several algorithms available to compute spherical Delaunay triangulations, however they all scale poorly with point size. To allieviate this issue, we are developing a parallel analog to Lloyd's algorithm which allows for parallel computations of spherical Delaunay triangulations.

Delaunay Triangulations

A Delaunay triangulation includes a set of Delaunay triangles of a point set. To be Delaunay a triangle must satisfy the following:

- Circumcircle must be empty
- At least 3 points lie on perimeter of circumcircle
- 4 or more points on perimeter means non-unique triangulation

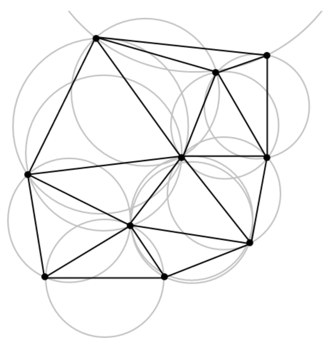


Figure 1: Delaunay triangulation, with circumcircles overlaid

Voronoi Diagrams

A Voronoi Diagram is the ortogonal dual mesh to a Delaunay triangulation. All Voronoi cells satisfy the Voronoi property given in equation 1.

$$V_i \in \hat{V} : V_i = \left\{ x \in \Omega | \|x - x_i\| < \|x - x_j\| \text{ for } j = 1, \cdots, n \text{ and } j \neq i \right\} \tag{1}$$

Figure 2 shows a Voronoi diagram overlaid on top of the Delaunay triangulation shown in figure 1.

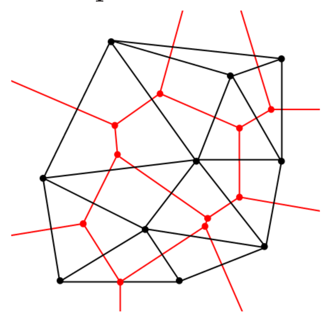


Figure 2: Voronoi diagram with corresponding Delaunay triangulation

Centroidal Voronoi Tessellations

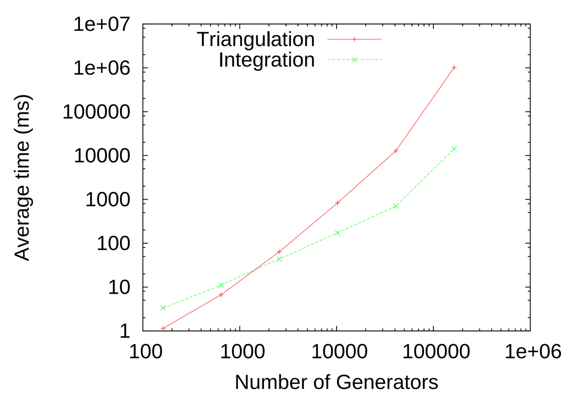
Typically, a Voronoi diagram is defined by a point set, known as generators. This Voronoi diagram is called a Centroidal Voronoi Tessellation when the set of generators of the Voronoi diagram is also the set of cell center of masses, as defined in equation 2.

$$\tilde{x}_i = \frac{\int_{V_i} \rho(x) x dx}{\int_{V_i} \rho(x) dx} \tag{2}$$

Where $\rho(x)$ defines a density function, which can control the resolution of the grid.

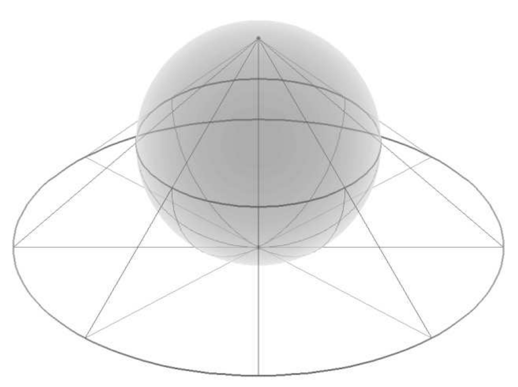
Motivation for New Algorithm

As the SCVT increases it's resolution, the triangulation (STRIPACK) cost becomes significantly more expensive than the integration step.



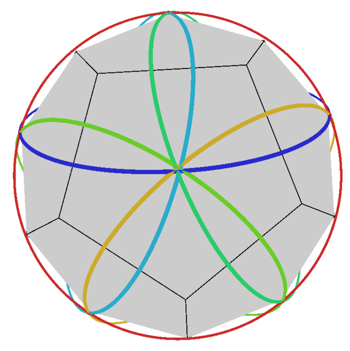
Stereographic Projection

Stereographic projections are conformal, meaning they preserve angles. In addition, stereographic projections have the unique property that they project circles and their interiors to circles and their interiors.



SCVT Based Domain Decomposition

In any parallel algorithm, some domain decomposition must be done to ensure load balancing and to help distribute the work to various processors. In this algorithm a coarse SCVT is used to define update boundaries, and various sorting algorithms can be used to determine which points each region is supposed to triangulate.



Non-Delaunay Triangles

A triangle that is non-Delaunay is easily defined as having the following criteria.

$$|\cos^{-1}(C_i \cdot c_j) + r_j| > R_i$$

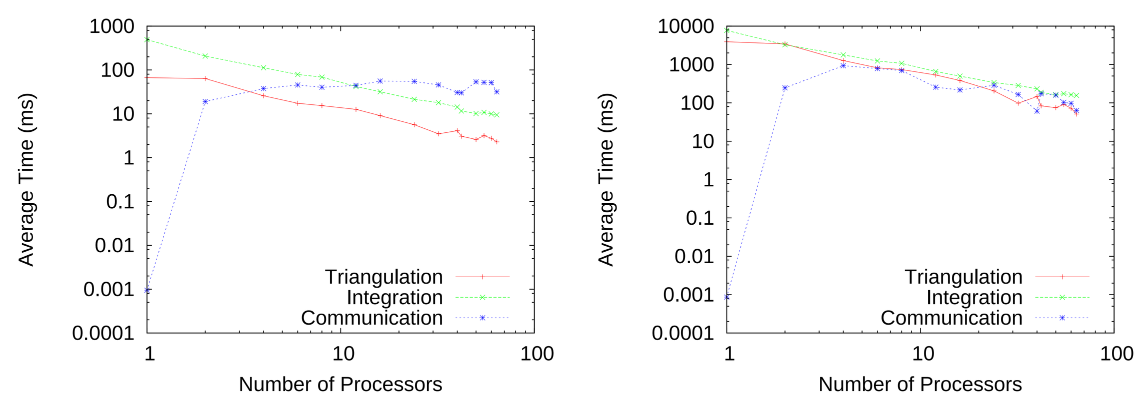
These triangles need to be removed from each regions planar triangulation.

Parallel Lloyd's Algorithm

Using coarse SCVT domain decomposition and stereographic projections, the parallel version of Lloyd's algorithm is as follows.

- Define starting point set
- Start Iteration Loop
  - Sort point into regions
  - Stereographically project points into planes
  - Triangulate points in planes
  - Map Triangulation back onto sphere
  - Remove Non-Delaunay triangles
  - Integrate Triangles to update point set
  - Check for convergence
  - Communicate updated point set
  - If Converged, stop, other wise loop again
- Print final point set, and triangulation

Quasi-Uniform Results

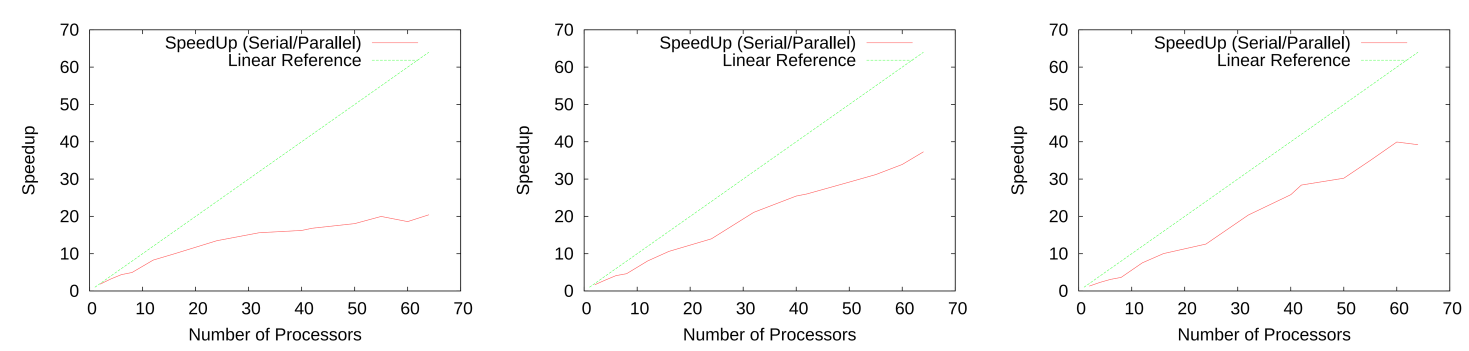


(a) 10242 Timing Results (b) 163842 Timing Results

Comparing serial STRIPACK to Parallel algorithm using Triangle for 163842 generators.

Algorithm	Procs	Regions	Time (ms)	Speedup
STRIPACK	1	1	207528.81	Baseline
MPI-SCVT IT	1	2	3623.09	57
MPI-SCVT F	1	2	9504.02	21
MPI-SCVT IT	42	42	50.6572	4092
MPI-SCVT F	42	42	5663.30	37

Parallel Performance Results



(a) 40962 Speedup (b) 163842 Speedup (c) 2621442 Speedup

Future Work

- Variable Resolution Results
- Performance Optimization
- Better Sort Algorithm

References

[1] P.L. Bowers, W.E. Diets, S.L. Keeling, Unpublished: http://www.math.fsu.edu/ aluffi/archive/paper77.ps.gz (2007)
[2] Q. Du, V. Faber, M. Gunzburger, SIAM Rev, **637-676**, 41 (1999)
[3] J.R. Shewchuk, Applied Computational Geometry: Towards Geometric Engineering, **203-222**, 1148 (1996)
[4] R.J. Renka, ACM Transactions on Mathematical Software, **416-424**, 23 (1997)