SELECTING INDEPENDENT VERTICES FOR TERRAIN SIMPLIFICATION

Bernd Jünger* **Jack Snoeyink†**
Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4 Canada

ABSTRACT
In this paper we investigate decimation algorithms for simplifying triangulated terrain models in order to support progressive transmission of GIS terrain models over the web. We report on experiments with heuristics that select an independent set of vertices to be deleted, while trying to preserve terrain characteristics.
Keywords: Level of detail, surface simplification, progressive transmission, accuracy, GIS, TIN.

1 TERRAIN MODELING

Digital Terrain Models or *DTMs* store, manipulate and display digital models of natural terrains for resource management (e.g. forestry) and development planning. They have important applications in *Geographic Information Systems (GIS)* and flight simulation. Mathematically, a terrain is the graph of a function defined on a region of the xy plane: $f: A \subset R^2 \to R$. In practice, we know the value of f (the elevation) for only a finite set of points on the earth's surface, gained by digitizing maps, remote sensing, etc. To assign elevations to points not in this set, we interpolate f.

There are three common representations for DTMs [Mar78]: a *Digital Line Graph (DLG)* stores polygonal lines that represent topographic contours, a *Digital Elevation Model (DEM)* stores a regular grid of elevation values, and a *Triangulated Irregular Network (TIN)* [PFLM78] stores a triangulated surface whose vertices are arbitrary sample points. Often the triangulation is based on the Delaunay triangulation in the xy projection, because of its well-shaped triangles.

By dispensing with the regularity of a grid, a TIN must store three coordinates for a vertex, as well as adjacencies between vertices. But because of its flexibility, it needs fewer vertices to approximate a surface at a certain "level of accuracy" than a DEM. There are many discussions whether DEMs or TINs are more appropriate representations of terrain [Kum94]; we focus on TINs because they are more amenable to simplification and display at multiple levels of detail.

An Application Scenario In particular, we address the following application scenario, suggested by Facet Decision Systems of Vancouver, Canada. Suppose that a server stores digital terrain models for various areas (*map sheets*) and clients request terrain models of these areas. If service is over a slow link to clients of varying capabilities and interests, as over the WWW, then the server would like to send models of various resolution, and do so progressively, so that new models are refinements of the old. (This not only reduces data transmission, but also helps preserve context from one model to the next.)

We implemented a prototype GIS client in Java, `http://www.cs.ubc.ca/spider/snoeyink/terrain/Demo.html`, based on [SvK97], in which the server computes an ordering of the vertices that supports fast reconstruction of the triangulated terrain on the client side using a simple, incremental algorithm. The prototype starts the reconstruction and visualization of a terrain while reading the input points from the server, and shows the current best approximation of the final result. The user can rotate the terrain and zoom in and out with the mouse; during this interaction the client shows a coarse model that still shows the characteristics of the terrain, but also provides good response time and smooth transitions between consecutive views, even on slow implementations of the Java runtime. As soon as mouse interaction ends, the most detailed approximation built so far is shown.

The implementation of the client as a Java applet makes it portable, and simplifies access to remote data.

---

*Supported in part by DAAD on an exchange with Technische Hochschule Darmstadt. bj@rbg.informatik.th-darmstadt.de

†Supported in part by grants from NSERC, the Killam Foundation, and CIES. Currently on leave at INRIA Sophia Antipolis, France. snoeyink@cs.ubc.ca

[SvK97] describe how to implement the *Painter's algorithm* for hidden surface elimination for terrains, since Java's 3-D graphics package is still under development.

2 TERRAIN SIMPLIFICATION

The goal of terrain simplification algorithms—or more generally: surface simplification—is to represent a surface with fewer vertices, but to keep a good approximation of the original surface. The advantages of working with a simplified version include reduced storage space, shorter times to construct the data structures, and faster visualization (rendering) of the surface. Many algorithms for terrain simplification were fueled by the need for DEM-to-TIN conversion. Such simplification algorithms usually aim for a certain approximation error with a small number of vertices, or a target number of vertices with a small approximation error.

We lack space for more than a brief classification of simplification approaches as they apply to our application. See the SIGGRAPH'97 course notes [Hec97] or the excellent survey [HG97] for more detail and references.

• **Refinement vs. Decimation** Refinement methods, for example Garland and Heckbert's Scape system [GH95], start with a coarse model and add vertices until a termination condition is met. Decimation methods, for example Lee's Drop Heuristic [Lee89], start with the finest resolution and discard points deemed unimportant. Refinement methods can attain higher accuracy with fewer points because they make global decisions on which vertex to add next—they pay for this in their speed, however.

• **One-pass vs. Multi-pass** One-pass methods assess the importance of vertices once to decide which to keep for the approximation; multi-pass methods, including our algorithm, work in phases and allow the importance of a vertex to change from phase to phase.

• **Sequential vs. Parallel** Sequential methods discard (or add) one vertex at a time, basing the next decision on a new approximation. Parallel methods discard (or add) a significant number vertices before computing the resulting approximation.

• **Preservation of original data** Some algorithms [Hop93, Hop96, Tur92] move vertices in order to get a more accurate approximation with a given number of vertices. Our algorithm, based on [SvK97], simply reorders the original data.

• **Feature orientation** Some of the first, and most successful, DEM-to-TIN conversions [FL79] were based on extracting features such as pits, peaks, valleys, and ridges. Ours does not do so, but could incorporate features into importance values assigned to vertices.

Note that in our application the approximations serve primarily as intermediate models during the transmission of data and the reconstruction of the complete triangulation. Thus, we focus on parallel decimation methods for their speed of computation.

3 DECIMATION BY DELETING AN INDEPENDENT SET

A key idea, which has long been used in theoretical work [Kir83], is that decimation by deleting an independent set of vertices—no two of which are joined by an edge—can be done efficiently in parallel: The deletions are independent and they leave one hole per deleted vertex, which can then be retriangulated independently. This decreases the programming complexity and running time. ([SvK97] shows that it can also decrease the time to rebuild the triangulation if the independent set is permuted to make locating the containing triangle easier.)

Since deletion and retriangulation is related to the degree of the vertices being deleted (usually $O(d^2)$ in the worst case), [Kir83, SvK97] advocate deleting low degree vertices, and prove that this leads to large independent sets ($\geq 1/6$ of the vertices). This approach ignores terrain characteristics, however. When visualizing the terrain during the rebuilding process, we want good approximations of the final result as early as possible. We therefore experimented with an alternative mechanism for finding independent sets that considers the characteristics of terrain: We apply a heuristic to assign an importance value to each vertex, then select an independent set to delete by greedily choosing vertices of low importance relative to their neighbors.

Before we describe the independent set selection algorithm and the heuristics to assign importance, however, we mention two other advantages of decimation by independent set: First, deleting an independent set in parallel reduces *drift*; this is in contrast to observations that refinement algorithms produce better approximations sequentially [GH95]. In a decimation algorithm, each operation produces an approximate terrain; for efficiency the criterion for the next operation is that it should approximate the previous approximation. Thus, globally significant features that are the sum of small contributions may disappear too quickly if a sequence of operations all err in a consistent direction. By deleting an independent set in parallel, drift is reduced—because the neighbors of a deleted vertex cannot be deleted, drift only occurs between phases, and there are only logarithmically-many phases.

Second, independent updates on rebuilding would be important for what Hoppe [Hop96] calls *geomorphs*—smooth transitions from a coarse to a refined mesh.

Morphing all points in an independent set produces approximately uniform change over a model, even at high resolutions.

3.1 Heuristics for vertex importance

The following heuristics assign an importance value to each vertex, based on its neighbors. High values are considered to represent high importance. Because a simple depth-first traversal is sufficient to reach every vertex of a triangulation, we focus on the computation of the importance value at a single vertex.

The *Degree method* of [SvK97], which ignores elevation information and choses vertices of low degree, is included in the experiments for reference.

Oblivious heuristics consider only the present triangulation and not the triangulation that will result from deletion of a vertex. We considered five variations:

The *Average Normal heuristic* (ANH) [And96] computes the negative of the length of the average of unit normal vectors to the faces surrounding a vertex. This value lies between -1 and 0, with -1 indicating a flat neighborhood. [SZL92] also weights each normal by the area of its face to minimize the importance of small ridges. We were surprised to find that this performed slightly worse, but do not include this variation here.

The *Height Difference heuristic* (HDH) sums the signed differences in elevation between a vertex and each of its neighbors, then takes the absolute value as the importance. A vertex has importance "zero" if its elevation could have been predicted by averaging the elevations of its neighbors.

The *Absolute Binary heuristic* (ABH) heuristic counts both the neighbors with a lower and a higher elevation and computes the importance of a vertex as the absolute difference of these counters. A relative variation (RBH) divides by the degree of the vertex.

The *Scape Order heuristic* orders the vertices by a sequential refinement algorithm like Scape [GH95] and assigns the rank of a vertex in this order to be its importance. Once assigned, these ranks never change. Note that our algorithm will usually generate approximations that differ from the ones obtained by Scape, because we delete an independent set and cannot simply delete the vertices rated least important by Scape.

Look-ahead heuristics do consider the change caused by deletion of a vertex.

The *Volume heuristic* measures the importance of a vertex by the absolute value of the change in volume that its deletion would cause. A volume of zero does not mean a flat area, but means that the amount of earth added to the terrain by the deletion balances the amount removed. The true symmetric difference would be more time-consuming to compute.

The *Drop heuristic* [Lee89] measures the vertical distance between a vertex and the linearly-interpolated elevation at the same x, y-coordinates if that vertex was deleted.

Analysis It is instructive to compare these heuristics on some examples, such as copies of a mountain peak at different scales or slants. For a 2-dimensional example, peaks u, v and w all have the same angle in figure 1, so ANH, ABH, and RBH do not distinguish between them. HDH and Drop do not distinguish between u and w. In 3-d examples, the Volume heuristic preserves volume but can change visual appearance.

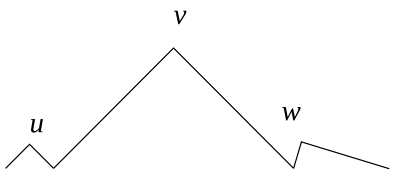

Figure 1: 2-d examples for heuristics

With respect to computation time, the oblivious heuristics are linear: Computing the importance of a single vertex takes time proportional to its degree; by Euler's relation the sum of all degrees is $O(n)$. Traversing the triangulation to visit all vertices is also $O(n)$.

For the look-ahead heuristics the running time can be of order $O(n^2)$ in the worst case. This is because each vertex has to be deleted to determine its importance. The deletion of a vertex of degree d takes time $O(d^2)$, so deletion of high-degree vertices is costly. Since high degree vertices are also bad for finding large independent sets, we can bound the degree to limit the asymptotic running time. The complexity and running time for look-ahead heuristics will still be higher, however.

It is natural to ask whether a combination of two heuristics can attain even better accuracy. We did not find combinations to give improvements; [GH95] had similar experiences with refinement methods.

3.2 Independent sets from importance

To construct an independent set from an assignment of importance values, it is natural to use a greedy strategy—go through the vertices in order of importance and take a vertex if none of its neighbors have been taken. As described, this would involve sorting vertices by importance or using a priority queue [And96]. In fact, we can compute the result of this greedy strategy in linear time without sorting.

The only data structure needed is a mark bit for each vertex; initially all vertices are unmarked. Using, say, depth-first search, visit every vertex of the triangulation. If a vertex v is unmarked, then find its status by the algorithm of figure 2 to determine if it is in the independent set.

```
findStatus(vertex) {
  if (marked(vertex)) return;

  nhbr := firstNeighbor(vertex);
  repeat {
    if (importance(nhbr) < importance(vertex)) {
      findStatus(nhbr);       /* recursion */
      nhbr := nextNeighbor(vertex);
    }
  } until (all nhbrs visited OR marked(vertex))

  if (not marked(vertex)) {
    addToSet(vertex);
    mark(vertex);
    for (every nhbr of vertex) mark(nhbr);
  }
}
```

Figure 2: Computing the status of one vertex

This algorithm checks that all neighbors of v that have lower importance are ineligible for the independent set because one of their neighbors has been chosen. (Recall that we want to select unimportant vertices to discard.) This check is done recursively, but it never follows an edge of the triangulation more than once because it always goes to vertices of lower importance and marks visited vertices on its return.

There is a potential flaw in our selection algorithm—a vertex becomes a member of the independent set with no regard to its importance when its neighbors cannot be members. To avoid this, we can use local importance—first marking vertices that are more important than a certain fraction f of their neighbors—or global importance—first marking the top $p\%$ of the importance values. The global has advantages that we can still prove that there will remain a large independent set and that linear time is sufficient.

4 EXPERIMENTAL RESULTS

In this section we present the experiments that compare how well the different ways of independent set selection preserve the shape of a terrain. We have space to report only a selection; more can be found in [Jue97].

To measure the quality of an approximation, we use the common evaluation method [Lee91, Kum94, GH95, And96] of computing the elevation difference between the approximation and the points not used as vertices. We evaluated several error measures (RMS, Mean Error, Maximum Error, 90%-ile error) and chose *Mean Absolute Error*—the average of the absolute values of elevation differences. Our tests showed that the ranking of heuristics according to their approximation error is mostly consistent over these error measures; but the measures are not equally apt to capture the "typical" error of an approximation, which we are interested in. RMS error, for example, emphasizes large differences, whereas the MAE weights all differences equally. We also employ visual comparisons to confirm the analytical results.

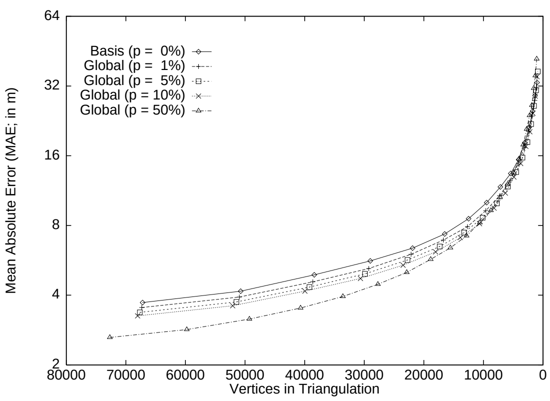

Figure 3: Retaining different percentages of globally important vertices

We use three different data sets for the experiments, which we refer to by size. The Crater Lake (Oregon, USA) data, used by [GH95] for their experiments with the refinement algorithm in the Scape system, contains about 154K points on a regular grid. The North Shore (Vancouver, Canada) data contains 88K points obtained from stereo orthophotos according to the BC TRIM standard: irregularly spaced but evenly distributed. We ran this data set through the refinement algorithm of Facet to obtian 44K points that were more unevenly distributed. All three data sets feature both mountainous and flat regions.

4.1 Keeping a percentage of important points

We first decide whether we should preserve the most important points—judged by local importance with fraction f or by global importance with percentage p—as suggested at the end of section 3.2. There are two indicators of effectiveness, which usually compete; on one hand we want to lower the approximation error, on the other hand we would like to keep the independent set sizes large. We show the comparison between the two strategies for one heuristic; tests with a other heuristics confirmed these results.

Figure 3 shows the Mean Absolute Error for the 88K file when $p\%$ of the points with highest global importance were preserved, for $p = 0\%, 1\%, 5\%, 10\%, 50\%$. Each curve shows one decimation process, with the points on the curves indicating phases determined by

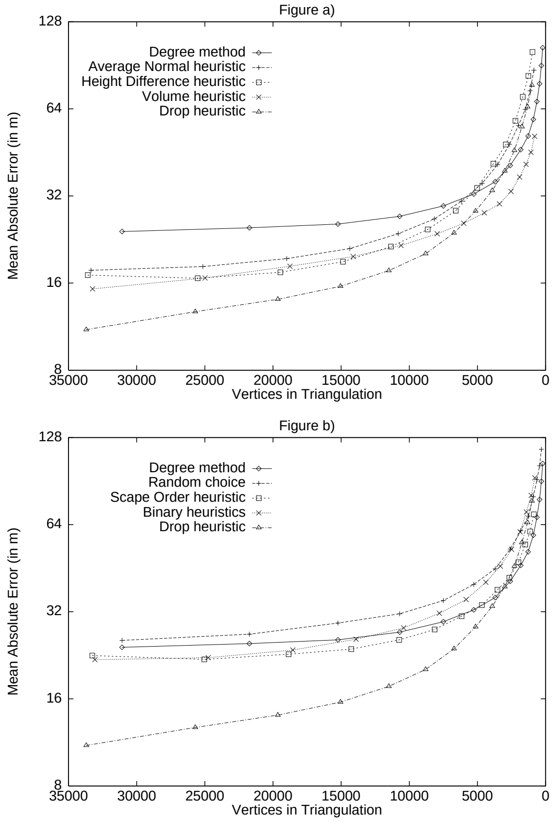

Figure 4: Results of terrain simplification heuristics (44K)

a single independent set at which MAE was measured. The leftmost point on each curve shows the vertices remaining after the first decimation, from which we can infer p's influence on independent set size. Thus, the fact that the curve for $p = 50\%$ starts to the left indicates a small independent set.

Experiments showed that keeping locally important points was inferior. For our subsequent experiments, we always keep the top $p = 5\%$ in global importance.

4.2 Analytical comparisons

Figure 4 shows the Mean Absolute Error for approximations of the 44K terrain. Each curve represents a decimation process with the respective simplification method. For clarity we break the comparison in two figures; each figure contains the curves for the Degree Method and the Drop heuristic as references, because they yield the least accurate and the most accurate approximations, respectively. Figure a) further illustrates the results for the Average Normal, the Height Difference and the Volume heuristics, which yield the best results besides the Drop heuristic. Figure b) shows the Scape Order and Binary heuristics, which generate the least accurate results among the terrain-preserving heuristics. Note that we show only one curve for the Binary heuristics, because the results for the Absolute

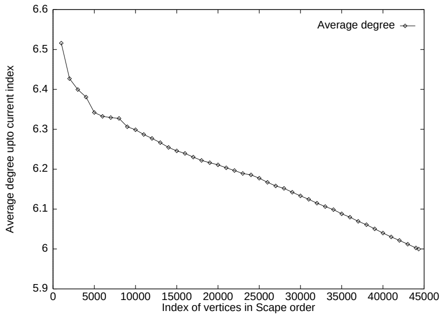

Figure 5: Relation between Scape order and vertex degree

and the Relative Binary heuristics are almost identical. In figure b) we also show the MAE for approximations that we obtain by randomly deleting vertices from the terrain without adhering to the scheme of deleting independent sets. For the random deletion we measured the MAE when the triangulation contained exactly as many vertices as after the phases of the Degree method.

In this experiment the Degree method generates surprisingly good approximations—consistently more accurate than random deletion, although it relies only on the degree of vertices and does not take any terrain characteristics into account. As the Delaunay triangulation depends only on the (x, y)-coordinates of the vertices and does not consider elevation information, we generally cannot expect a correlation between the degree of a vertex and its importance for the terrain. Thus, with respect to the terrain characteristics, the Degree method should be a random process, too.

One possible reason why the Degree method performed better than random deletion is that the 44K file had been obtained by simplifying a larger data set with Garland and Heckbert's Scape [GH95] algorithm. We found a definite correlation between vertex degree and the Scape order, shown in figure 5. Each point on the curve stands for the average degree of vertices up to the respective index.

The results for the 88K and 154K files are given in figure 6. In these files, all terrain-preserving heuristics perform consistently better than the Degree method. There remains a gap between the random and Degree methods, which requires an additional explanation for these files, especially since the 154K file is a grid with no correlation between degree and importance. A possible explanation is that the sequential random deletion suffers more from *drift* (section 3) than deletion in parallel by degree.

In general, the tests with the two larger files paint a more consistent picture of the methods we compared than the test for the 44K file. The ranking of heuristics stays the same over the complete decimation processes. The Drop heuristic again yields the best results. The

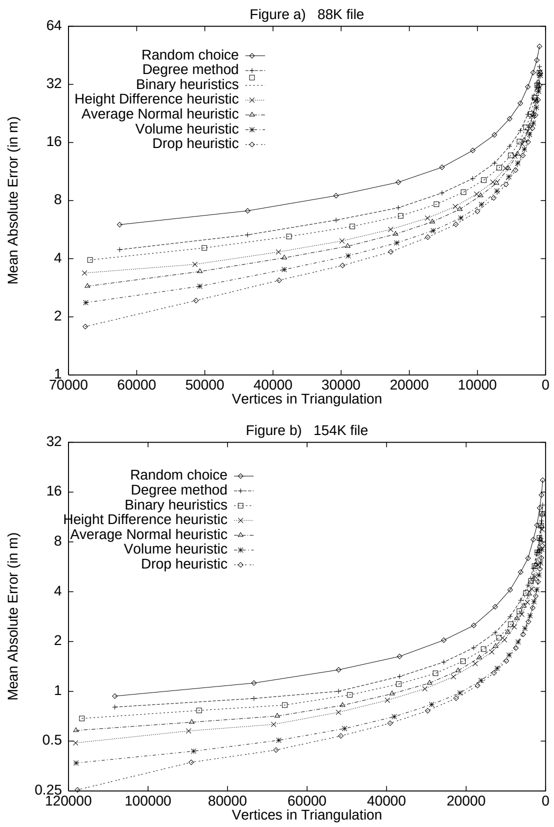

Figure 6: Heuristics on 88K and 154K files

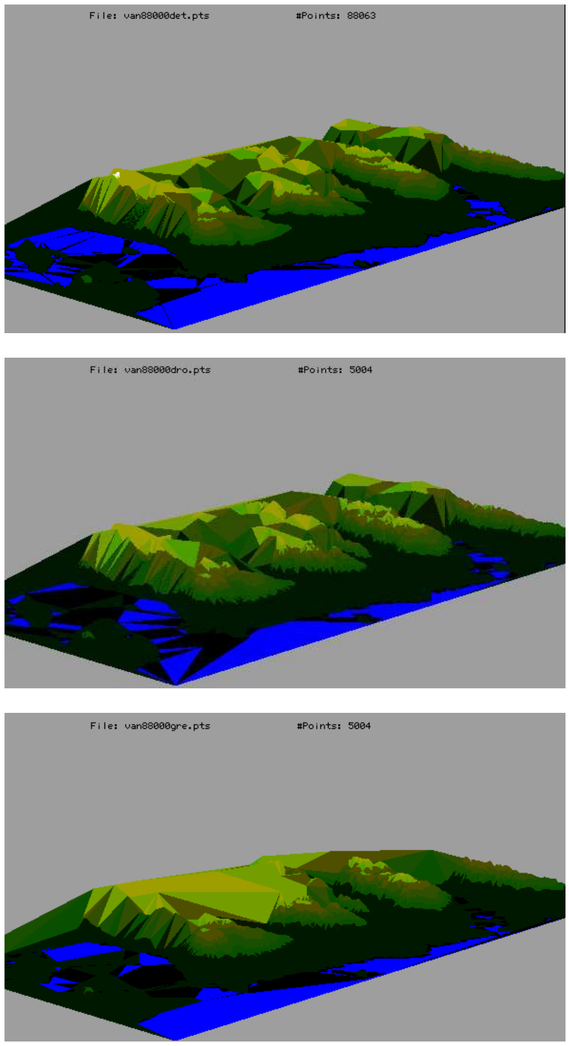

Figure 7: Top to bottom: 88K terrain, Drop heuristic, Degree method (both 5000 vertices)

only major difference between the results for the two data sets is that the Height Difference and the Average Normal heuristics change ranks. Note that the improvement of the Degree method over random deletion is higher than the improvement of some of the heuristics over the Degree method. This becomes more evident when we use a non-logarithmic scale for the y-axis. But then the curves for the terrain-preserving heuristics move too close to each other to be discernible.

In both figures 4 and 6 we perceive that the independent set sizes for terrain-preserving heuristics are smaller than for the Degree method. Among the heuristics the difference in independent set sizes is not significant. The Binary heuristics seem to produce the largest independent sets, but at the cost of poor approximations.

4.3 Visual comparisons

We visually compared the approximations with different heuristics, to find out whether their appearance confirmed the analytical result. We modified our Java GIS client to view several approximations—and optionally the complete terrain—in different displays simultaneously. The displays are coupled for rotating or zooming, so that we look at each approximation from the same viewpoint. The application starts with only the bounding box vertices for each approximation; on demand it adds a certain number of vertices to each.

We compared the outcomes of the Degree method, the Drop, the Volume, and the Height Difference heuristics. We include one example. Figure 7 shows (from top to bottom) the complete 88K terrain and approximations obtained with the Drop heuristic and the Degree Method. Both approximations contain 5000 vertices, which is about 6% of the total number. Some differences can be seen in the leftmost mountain range, where the Degree method has eliminated more detail from the mountainside.

The visual results were not always as clear as looking at the Mean Absolute Errors would imply. The Height Difference and Volume heuristics sometimes appeared closer to the complete terrain than the Drop heuristic. This lets us conclude that either the elevation difference as an error measure favors the Drop heuristic, because they share the same criterion, or that smaller differences in the error measure are not necessarily sig-

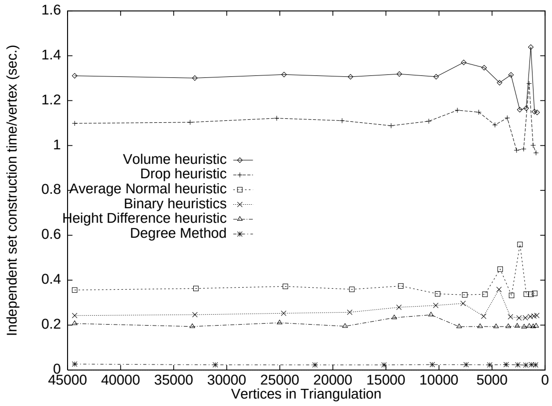

Figure 8: Independent set construction times

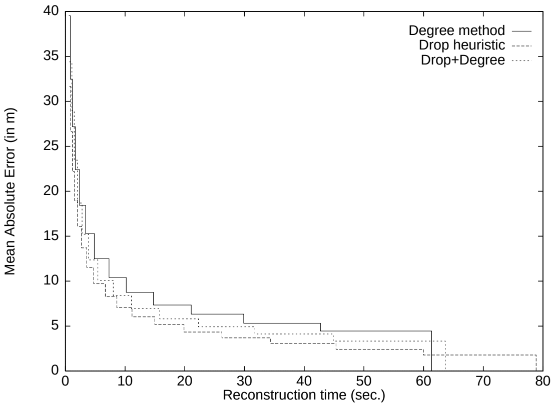

Figure 9: Accuracy versus reconstruction time

nificant. In general, however, the differences between the three terrain-preserving heuristics were not large. The our comparisons illustrated the following trends:

- Terrain-preserving heuristics yields good approximations early in the rebuilding process.
- For the 44K file, which had been simplified with Scape, the Degree method also yields early approximations with good appearance. This observation supports the analytical results in figure 4.
- For the files that have not been simplified before, the terrain-preserving heuristics have a significant edge over the Degree method.

4.4 Running times of simplification methods

When we consider the running times of the methods, there are two times to be concerned with: Time taken by the server to compute a simplification is less important because it can be done once and the resulting ordering of vertices can be stored. Time taken by the client is harder to measure because of the tradeoffs between time and accuracy.

Thus, figure 8 shows the time the server took to construct independent sets. Times were measured by Java's *System.currentTimeMillis()* method, using the runtime environment for Sun's beta Just-In-Time (JIT) compiler on a 170MHz ultraSparc.

We can identify three categories: the fast Degree method, the medium-range oblivious heuristics, and the complex look-ahead heuristics. Thus the effort for importance determination roughly mirrors the approximation accuracy that we obtain with the different methods. When we compare representatives of different categories, the more complex methods yield more accurate results. Within the categories this is not necessarily the case.

To compare client times, we graph in figure 9 the MAE of the Degree and Drop heuristics as a function of time to reconstruct the corresponding model on the client end. This figure also includes a Drop+Degree heuristic that we describe below. We chose a stair to represent the approximation errors, instead of interpolating the data points with smooth curves; this better illustrates the impression that a user gets during the incremental visualization of a terrain, because the visible image does not change during one phase of reconstruction.

We see that the approximations obtained with the Drop heuristic have a lower error throughout the reconstruction, up to the point where the reconstruction for the Degree method finishes and is able to show the terrain in full detail. From this point the reconstruction for the Drop heuristic needs 17 more seconds to finish. The decision between the Degree method and terrain-preserving heuristics thus involves a time versus accuracy tradeoff: network and and drawing speed will also play a role in determining which is better.

Combining Drop and Degree It is once again natural to ask whether a combination of heuristics provides better time/accuracy tradeoffs. There are two easy ways to combine the Degree and Drop methods: in *Degree+Drop* we use the Degree method, but appeal to the Drop heuristic whenever there is a choice of vertices of lowest degree, and in *Drop+Degree* we make the $p\%$ most important vertices according to the Drop heuristic ineligible for the independent set, and then apply the Degree method to the remainder.

In figure 10 Drop+Degree is more accurate, so we included it in figure 9. The running time for Drop+ Degree is only slightly larger than for the Degree method, but the approximation errors are lower except for short periods early in the reconstruction.

5 CONCLUSION

We have investigated a number of simplification heuristics to support progressive display of TIN terrain models in GIS under a scenario in which a server transmits

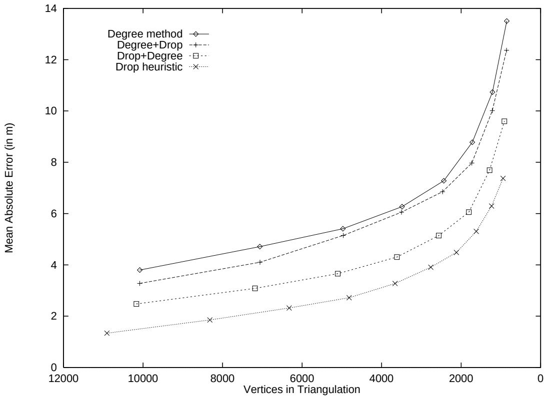

Figure 10: Combining Degree method and Drop heuristic

the data for a client to reconstruct the terrain model. Choosing independent sets makes the client's task easier and has other advantages.

Employing terrain-preserving heuristics to choose the independent set improves the quality of approximations that the client meets during the reconstruction process. The Drop and Volume heuristics yield the best results, but are relatively expensive because of their look-ahead criteria. Height Difference and Average Normal heuristics work faster and produce acceptable results. The other heuristics tested were slower or poorer, or both.

The Degree method, which ignores elevation, gives surprisingly good approximations when the input data has been simplified with a refinement algorithm like the Scape [GH95]. In general, its results are far better than when we delete the same number of vertices randomly.

We were able to combine the Degree method and the Drop heuristic (or any of the heuristics) to produce independent sets that are almost as large as those of the Degree method, with approximations of higher quality. We think that this method is the most promising of the techniques that we presented.

ACKNOWLEDGEMENT

We thank Facet Decision Systems for the use of their system and access to terrain data.

References

[And96] D. S. Andrews. Simplifying terrain models and measuring terrain model accuracy *Msc. Thesis. Tech. Report 96–05, Dept. of CS, UBC*, 1996.

[FL79] R. J. Fowler, J. J. Little. Automatic extraction of irregular network digital terrain models. *SIGGRAPH '79*, 199–207, 1979.

[GH95] M. Garland, P. S. Heckbert. Fast polygonal approximation of terrains and height fields. *Carnegie Mellon Tech report CMU-CS-95-181*, 1995.

[HG97] P. S. Heckbert, M. Garland Survey of Polygonal Surface Simplification Algorithms See also [Hec97]

[Hec97] P. Heckbert, organizer. Multiresolution Surface Modeling. Course 25, *SIGGRAPH '97.* `ftp://ftp.cs.cmu.edu/afs/cs/project/anim/ph/paper/multi97/release/intro.html`

[Hop93] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle. Mesh Optimization. *SIGGRAPH '93*, 19–26, 1993.

[Hop96] H. Hoppe. Progressive Meshes. *SIGGRAPH '96*, 99-108, 1996.

[Jue97] B. Jünger. Terrain modeling as an application of a fast reconstruction algorithm for Delaunay triangulations. *Tech. Report 97–8, Dept. of CS, UBC*, 1997.

[Kir83] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12:28–35, 1983.

[Kum94] M. Kumler. An intensive comparison of Triangulated Irregular Networks (TINs) and Digital Elevation Models (DEMs). *Cartographica*, 31(2), 1994.

[Lee89] J. Lee. A drop heuristic conversion method for extracting irregular network for digital elevation models. *GIS/LIS '89 Proceedings*, volume 1, 30–39, 1989.

[Lee91] J. Lee. Comparison of existing methods for building triangular irregular network models of terrain from grid digital elevation models. *International Journal of Geographical Information Systems*, 5(3):267–285, 1991.

[Mar78] D. M. Mark. Concepts of "data structure" for digital terrain models. *Proceedings of the Digital Terrain Models (DTM) Symposium*, 24–31, 1978.

[PFLM78] T. K. Peucker, R. J. Fowler, J. J. Little, D. M. Mark. The triangulated irregular network. *Proceedings of the Digital Terrain Models (DTM) Symposium*, 516–532, 1978.

[SvK97] J. Snoeyink, M. van Kreveld. Linear-time reconstruction of Delaunay triangulations with applications. *Algorithms — ESA '97*, Springer-Verlag, New York, in press. `http://www.cs.ubc.ca/spider/snoeyink/papers`

[SZL92] W. J. Schroeder, J. A. Zarge, W. E. Lorensen. Decimation of Triangle Meshes. *SIGGRAPH '92*, 65–70, 1992.

[Tur92] G. Turk. Re-Tiling Polygonal Surfaces. *SIGGRAPH '92*, 55–64, 1992.