Gradient Error and Origami

My Work With The Space Time Programming Group

Josh Horowitz

MIT CSAIL

November 2, 2007

UROP Proposal

Two branches of research:

1. Analyze errors of gradients.

   '...I will investigate error in the ubiquitous "gradient" algorithm, which determines the distance from a processor to a designated region. My investigation will involve both conceptual analysis of mathematical models and quantitative analysis of computer-run simulations.'

2. Make Proto fold.

   '...I will also pursue the project of implementing Radhika Nagpal's biologically-inspired Origami Shape Language in the STPG's Proto programming language.'

The Algorithm

The gradient algorithm gives a way to estimate global distance (across many radio ranges) from local distance.

1. Initialize the source S at 0 and all other nodes at ∞.
2. Continually update the gradient value of all nodes $N \notin S$ based on their neighbors in order to keep the triangle inequality maintained:

$$\mathrm{Gradient}(N) = \min_{N' \text{ near } N} \left(\mathrm{Gradient}(N') + d(N, N') \right).$$

Essentially, this gives the length of the shortest path from N to the source in a graph with edges connecting nearby nodes.

The Problem

Even with error-free processing and error-free local range-finding, the gradient algorithm is not error-free: trails are jagged.

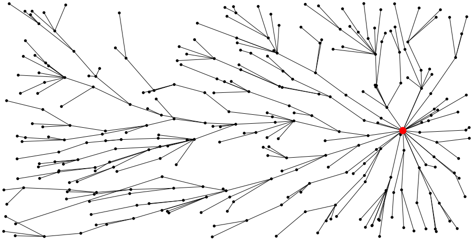


Figure: Gradient trails back to a source.

- This jaggedness introduces both systematic and statistical error into the gradient algorithm.
- I studied one particular kind of systematic error, the fact that the gradient algorithm more accurately measures $\alpha d(N,S)$ than $d(N,S)$ itself (for some constant $\alpha \gtrsim 1$).
- That is, there is an underlying jaggedness that asymptotically causes gradient values to be scaled upwards by a constant factor α.

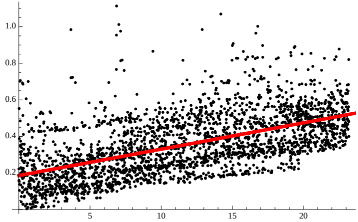


Figure: $d(N,S)$ vs. $[\mathrm{Gradient}_S(N) - d(N,S)]$ (so the red line has slope $\alpha - 1$)

- There are different ways of looking at this phenomena:
  - It is unfortunate error which we would like to avoid. We need to know how high node density (N_{loc}) has to be to put α close enough to 1 for our purposes, whatever they may be.
  - More reasonably: It is a feature of how the gradient algorithm works which we would like to correct for. We need to know what α is in terms of N_{loc}.
- Either way requires knowing the relationship between N_{loc} and α.
- I first investigated this question through direct experimentation.

Basic Setup

- Fill a rectangle randomly with nodes.
- Set a thin strip along the left edge to be the source.
- Run a gradient.

The output of a run will be a set of pairs $(d(N,S), \mathrm{Gradient}_S(N)) = (\text{actual distance}, \text{calculated distance})$. We perform a linear regression to determine the value of α for this particular run. After many runs, with varying N_{loc}, we will have a large set of $(N_{\mathrm{loc}}, \alpha)$s to analyze.

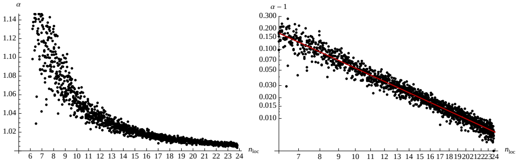


Figure: N_{loc} vs. α (linear and log-log).

Initial Results

When I gave my first presentation on my research, things seemed to be pointing nicely to a $\alpha \propto {n_{\text{loc}}}^{-2}$ relationship. The two experiments represented below gave fits $\alpha = 0.693{n_{\text{loc}}}^{-2.212}$ and $\alpha = 0.648{n_{\text{loc}}}^{-2.045}$, respectively.

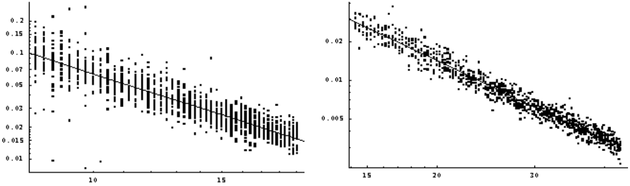


Figure: Old-school data.

Condor

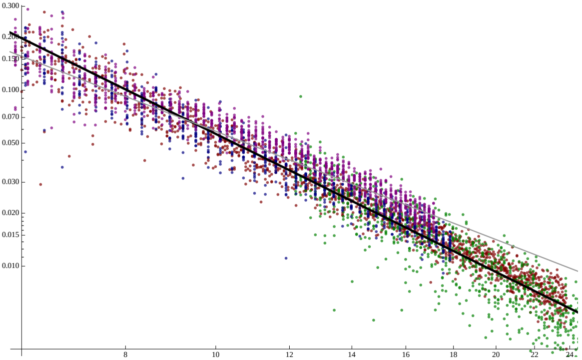
To continue exploring the parameter space effectively, I developed a system to run Proto simulations on CSAIL's Condor-based computing cluster. This required:

- Stripping Proto of its graphics code, so it can compile and run on the cluster.
- Figuring out how to make Condor play nice with output of dump files.
- Making scripts to generate large "submit" files for Condor.

Newer Results

With the computing cluster, I could run Proto thousands and thousands of times with thousands of nodes per run. The results threw doubt on the simplicity of a -2 exponent:

Each point is a run: they are partitioned into four experiments (red, blue, green, and purple). The dark black line is the fit, $\alpha = 22.91 \cdot n_{\text{loc}}{}^{-2.608}$. The pale line is the fit from the old set of experiments.



I took a break from running experiments and analyzing data to see if I could derive an expression for α purely from theory.

Theoretical Model

- Consider a node N at $(0,0)$, with the gradient source infinitely far to the right (positive x direction).
- If N uses some neighboring node N_1 located at (x,y) in its gradient path to the source, it will move us x closer to the source, while increasing the gradient value by $\sqrt{x^2+y^2}$.
- If this pattern continues, we will have $\alpha = \frac{\sqrt{x^2+y^2}}{x} = \sqrt{1+\left(\frac{y}{x}\right)^2}$.
- Assuming N_1 is distributed uniformly in the half-circle to N's right, a simple geometric argument yields the cumulative distribution $F(A) = P(\alpha < A) = \frac{2}{\pi}\arctan\sqrt{A^2-1}$. Taking the derivative of this yields the probability density function $f(\alpha) = \frac{2}{\pi}\frac{1}{\alpha\sqrt{\alpha^2-1}}$.

- Now suppose N has exactly $N_{\mathrm{loc}}/2 = n$ neighbors to its right.
- Each gives rise to an α with an identical distribution to that just derived (That is, we suppose that we have $N_{\mathrm{loc}}/2$ independent and identically distributed random variables). To calculate the distribution of their minimum, we use the order-statistic formula:

$$f_{\min}(\alpha) = n(1 - F(\alpha))^{n-1} f(\alpha) = n\left(\frac{2}{\pi}\right)^n \frac{(\operatorname{arccot}\alpha)^{n-1}}{\alpha\sqrt{\alpha^2 - 1}}.$$

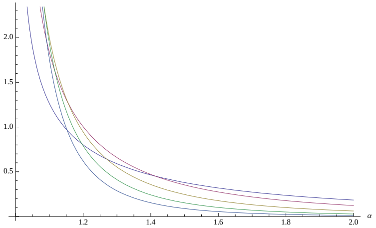


Figure: Plots of $f_{\min}(\alpha)$ for $n = 1, 2, 3, 4, 5$ (higher $n \implies$ lower $f_{\min}$ as $\alpha \to \infty$).

- The expected values of these distributions are

$$E(\alpha_{\min}) = n\left(\frac{2}{\pi}\right)^n \int_0^{\pi/2} \beta^{n-1} \csc\beta \, d\beta.$$

Mathematica can't touch these, but we can analyze them numerically.

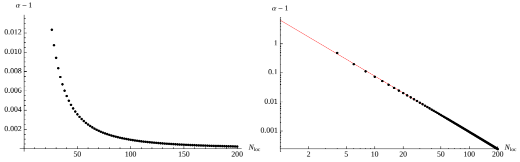


Figure: n_{loc} vs. α (linear and log-log).

- The fit line plotted in the second graph is $\alpha = 6.476{n_{\text{loc}}}^{-1.922}$.

Theory vs. Experiment

- Predicting an exponent of -2 used to sound good, but more thorough experimentation seems to suggest that it is innacurate.

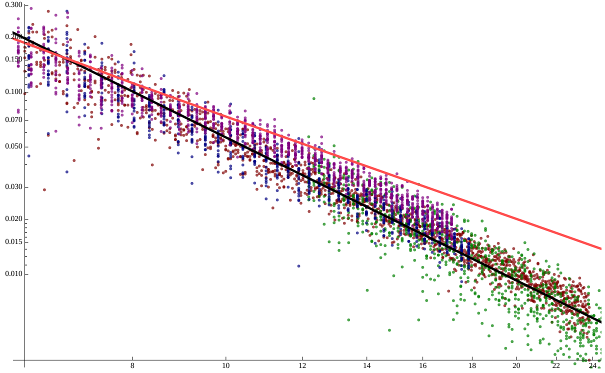


Figure: n_{loc} vs. α (red line is the prediction from theory).

Theoretical Model

Where does this discrepancy come from?

Assumptions

1. The source is very far away.
2. There is no variation in the number of neighbors.
3. The distribution of neighbors of any two nodes is independent (even for nodes with overlapping neighborhoods).

None of these holds exactly during the execution of a gradient. It is unknown whether their failures invalidate the results completely, introduce additional phenomena to augment the results with, or are completely negligible.

Origami with Proto

- In her PhD thesis, Radhika Nagpal describes a foldable sheet of cells which, purely through local interaction and actuation, can attain a shape determined by a global specification.
- My other goal for the summer was to implement this in Proto. This consisted of several tasks:
    1. Implement a folding mechanism in the Proto simulator and add language hooks to access the new features (actuators and sensors).
    2. Figure out how to determine creases and sequence folds in Proto code.
    3. Write a program to transform a specification written in Nagpal's high-level Origami Shape Language into a Proto program which folds the material as specified.
- I finished the first two of these to some degree.
- Unfortunately, I can't get it to run right now. :-(

Conclusion

- I hope that my work has been of value (perhaps I should finish it off and write it up to make sure. . .).
- Especially looking back, it is clear that there are many interesting questions left to explore.
- This summer was one of the best in my life. Thanks to everyone I worked with for their help, encouragement, and patience.