

## **Redistricting Utopia**

Today gerrymandering is a serious threat to democracy in America and other countries around the world. When a single party wields enough power to force their own set of districts on the public, then they can ensure that their party will maintain power for a very long time. Most states do a poor job of defining what fair districts are and leave it to the courts who usually interpret a fair district as compact and having good apportionment (the concept of “one person, one vote”). This leads to partisan politicians defining both terms in the ways that suite them best. Apportionment is relatively easy to handle because you just have to have roughly the same number of people in each district. Compactness, however, is much more difficult to formalize. Last summer, we proposed the gerrymander factor as a measure of compactness and defined it as the square of the perimeter of the district divided by its area. The work we did last summer has shown that this is an effective measure of compactness and using it can lead to surprisingly good results. Therefore, the problem becomes finding the best weighted sum of our measures of compactness and apportionment.

In our previous research, we used a genetic algorithm to solve this problem. It started with many, random redistricting plans, each represented as a string of 1s and 0s that is in many ways analogous to DNA. A process based on natural selection was then used to generate new redistricting plans from several promising plans and to remove bad plans from the pool of plans under examination. Natural selection, of course, is based on survival of the fittest. We measured the fitness of a redistricting plan as a combination of how evenly the population was spread between the districts (to meet the apportionment requirement) and how small the gerrymander factor of the districts was. This approach led

to amazingly good results that were in many cases much better than the actual districts used in the Seattle area.

We would like to continue our research this summer by applying the experience we gained last summer and the knowledge we have gained from further investigation during the school year to refine our genetic algorithm in the hopes of producing better results more quickly. We believe we can do this by reimplementing the algorithm so that the code is cleaner and more efficient and by using fitness-proportional selection instead of rank-proportional selection. Rank-proportional selection occurs when the redistricting plans are listed in order of fitness values and plans near the top of the list are given preference over plans near the bottom of the list when choosing plans for reproduction. Fitness-proportional selection bases the decision on the fitness rating for each plan rather than just the rank. For instance, if the first and second ranked plans had fitness values of 100 and 102, then fitness-proportional selection would be about as likely to pick one as the other, whereas rank-proportional selection would be much more likely to pick the plan ranked at 100 because it is ranked number 1 instead of number 2. We believe that fitness-proportional selection is more representative of the process that actually occurs in nature and is therefore likely to provide better results. Reimplementing the algorithm will also allow us to make the program more modular so that it is easy to adapt other algorithms to solve the redistricting problem. Some other algorithms we intend to investigate are other flavors of genetic algorithms and simulated annealing.

Last summer we experimented with a cluster-computing approach to running our algorithm. We loaded our program onto 8 computers and let them all attempt to find good solutions while sharing their progress with each other periodically. By allowing these computers to work together on the problem, we can dramatically increase the likelihood of

finding a good redistricting plan. Since we will be rewriting significant portions of the program anyway, it would give us a chance to refine and extend the cluster-computing code to create a more robust cluster of computers. We would do this by allowing computers to dynamically join in on solving the problem and to drop out of the cluster with minimal impact on the computations. This would allow us to leverage the computing power of the 30+ computers in the computer science lab since they could be used for problem solving while idle, but would remove themselves from the cluster if someone needed to use them for class or homework. With so much more computing power and an updated version of our code, we believe we can find even better redistricting plans than we were able to last year.

Secondly, we hope to finish the process of importing the state of Washington's GIS data so that our program can be tested by creating redistricting plans for various levels of governments in Washington. We started on this last summer by examining ways to deal with rivers, lakes, and other bodies of water since they can pose problems for our genetic algorithm. Another problem we discovered is that the state's data is a combination of data from each county and that the counties often do not agree on their own boundaries. Therefore, we will need to find a way to fix the data from each county so that state-wide redistricting plans can be created. Last summer we started creating a computer program that was capable of doing these things and we hope to finish it this summer. We would then be able to import not only Washington State's data, but the GIS data from any state. We could then compare our algorithm's output against the gerrymandering that was done in Texas and other states recently

Finally, we hope to include a second student in our research. This new member would analyze the redistricting plans we create by examining if and how our plans would

have affected previous elections. They will also consider other factors that might be taken into account when redistricting plans are made. They would then attempt to formalize these factors so that they could be included in the program we are creating.

A note on copyright: Although the code developed for this project may be regarded as 'work for hire' under copyright law, I propose that UPS assign to me all rights to code I write for this project, provided I make the code available to everyone under the GNU General Public License or comparable open source license. The University of Puget Sound would maintain copyright on all other works produced as part of this research