AC290 EXTREME COMPUTING: REPORT 2

Bleeding Edge

Gioia Dominedo, Christian Junge, Abhishek Malali and Isadora Nun

October 7, 2015

1 What is the problem?

The Influence Maximization Problem examines the behavior of nodes of a graph influencing one another. For this assignment, we are looking at a graph of 240 nodes (representing individual Yelp reviewers) and the connections (edges) between them, and examining how the nodes within this graph activate one another. The goal is to find the optimal set of starting nodes, that is, the set with the highest average number of nodes activated when the simulation is run repeatedly.

2 What is independent cascading?

In the independent cascading approach, one or several nodes begin the simulation as activated. In each iteration of the algorithm, each activated node has one chance to activate the nodes that are adjacent to it. The probability that an activated node (v) will activate a node adjacent to it (w) is given by the Beta function with the parameters α and β, where α is the number of reviews that reviewer w has written after reviewer v (within a time frame given by τ), and β is the total number of reviews that reviewer w has written. Within the algorithm, this is implemented by selecting a random number from the Beta distribution and a random number from the Uniform distribution, and if the number from the Uniform distribution is less than the number from the Beta distribution, the node is activated. Each node gets only one chance to activate its non-activated neighbors, and the independent cascade algorithm terminates when there are no more possibilities to activate further nodes. The algorithm returns the total number of nodes that have been activated in the simulation.

3 What is the Influence function?

The influence function is represented as follows:

$$f(S) = \frac{1}{N} \sum I(S) \tag{1}$$

Where N is the number of simulations, f is the Influence function, I is the independent cascade function, and S is a given set of starting nodes. The influence function tells the average number of nodes activated by a given set of starting nodes when the simulation is run N times, in other words the expected value of the number of activated nodes. In the Influence Maximization Problem, we would like to find the set S of starting nodes that yields the highest value of the Influence function.

4 What is Greedy algorithm?

The greedy algorithm runs a given number (N) of simulations, given a number of starting nodes. In its first iteration, it finds the single node that has the highest expected value of activated nodes when run over N simulations. In each subsequent iteration, it finds the next node which, when added to the starting set, returns the highest expected value of activated nodes when run over N simulations. In this way, additional starting nodes are incrementally added to the set which was previously determined to be the best set until the set contains the desired number of starting nodes.

The danger in this approach is that a set of n starting nodes may be the best n-element sized set of starting nodes, but the best set of nodes that is created by adding one more element to size-n set may not be the best size(n+1) set possible.

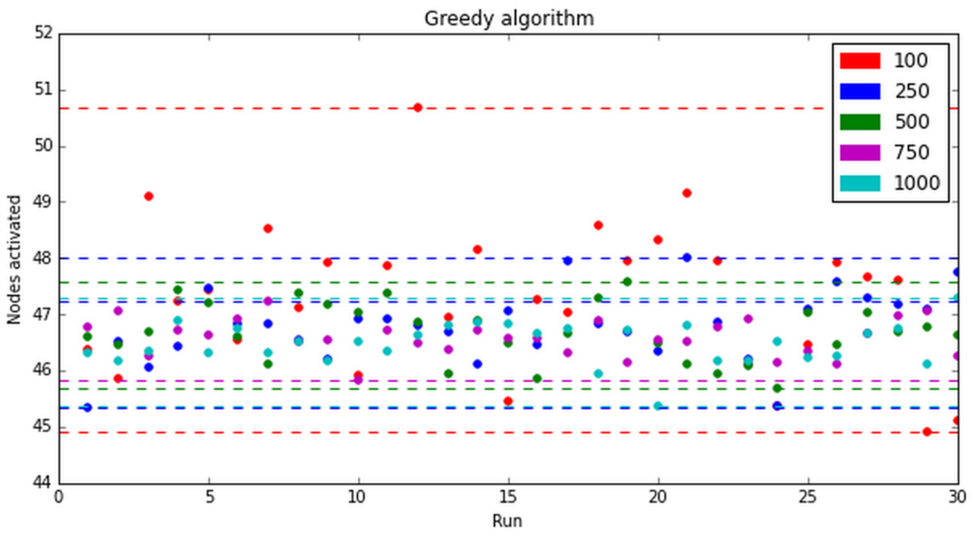

Figure 1: Plot of minimum and maximum activations for different number of runs

As we can see in figure 1, we observe that with an increase in the number of the runs, the maximum and minimum activations bounds become tighter and tighter. The minimum bound also increases as expected with the increase in the number of runs for the greedy algorithm.

5 What is Simulated Annealing?

Simulated Annealing is a process to stochastically find the optimal starting set of nodes. This optimization process is used for discrete search spaces where an exhaustive search would take a prohibitively long time. In the case of the Maximum Influence Problem, testing all possible combinations of nodes would take a very long time, so simulated annealing is a sensible approach. For this problem, the goal is to find the set of starting nodes with the highest value of the influence function; in other words, the set of starting nodes with the highest expected number of activated nodes.

The process works as follows: First, an initial set is selected and the expected value of activated nodes for a given number of simulations is calculated. Then this starting set is perturbed (for example, by switching some of the starting nodes), and the expected value of this new set is calculated. If this new set yields a higher expected value, then it is accepted, but if this new set yields a lower expected value, then it is accepted with a probability based on the temperature ($P(X = j) = \exp(\frac{-\Delta E}{T})$). This process runs over many iterations, yielding a set that returns the highest expected value of activated nodes.

In the simulated annealing process as in the greedy algorithm, there remains the potential that the actual optimal solution is not found because the set is “stuck” with some nodes that yield good intermediate results when included in the set, but are not the best overall. For this reason, the concept of “heat” is introduced into the annealing process. This works as follows: If the new starting set returns an expected value that is much higher (“higher” is relative, and can be calibrated in the algorithm), then the number of starting nodes that are switched in subsequent runs of the influence function is kept the same (or potentially increased). This is referred to as “heat”, and determines how drastic of a change is made to the considered set of starting nodes at each step in the simulated annealing process. In contrast, if the expected value is not much higher, the algorithm “cools”, and fewer nodes are switched. Periodically, the algorithm is “heated up” to stimulate it to make more drastic changes to ensure that it is not stuck in a relative maximum that is lower than the absolute maximum.

Over many iterations, the simulated annealing process should yield the set of points with the highest expected value of activated nodes.

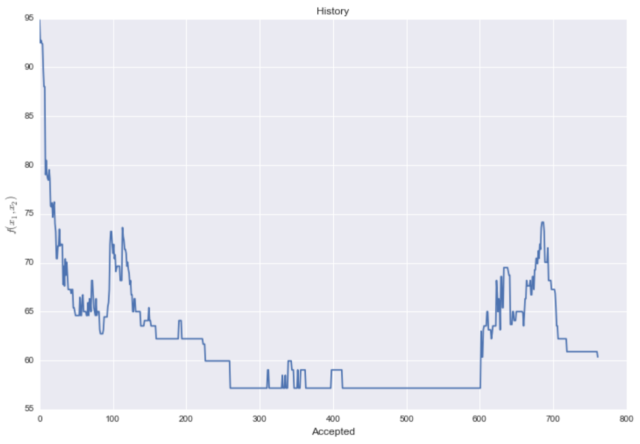

Figure 2: Simulated Annealing

6 Uncertainties in the optimization

Uncertainty in the optimization results from stochastic approaches, which yield estimates with an unknown amount of error rather than definite values. In particular, the influence function returns an expected value that has a certain amount of uncertainty. While the standard error of the independent cascade function is endemic to a given set of starting nodes, regardless of the number of simulations, the standard error of the Influence function shrinks as the number of independent cascade simulations (N) increases. Therefore, a larger number of simulations will yield a more accurate estimate of the average influence for a given set. We see that this tightening of the standard error is proportional to $\frac{\alpha}{N^\lambda}$. By fitting the data with a linear regression we obtain $\frac{11.34}{N^{0.46}}$ So any optimization algorithm using the influence function will be impacted by its inherent uncertainty, which can only be reduced by increasing the number of simulations.

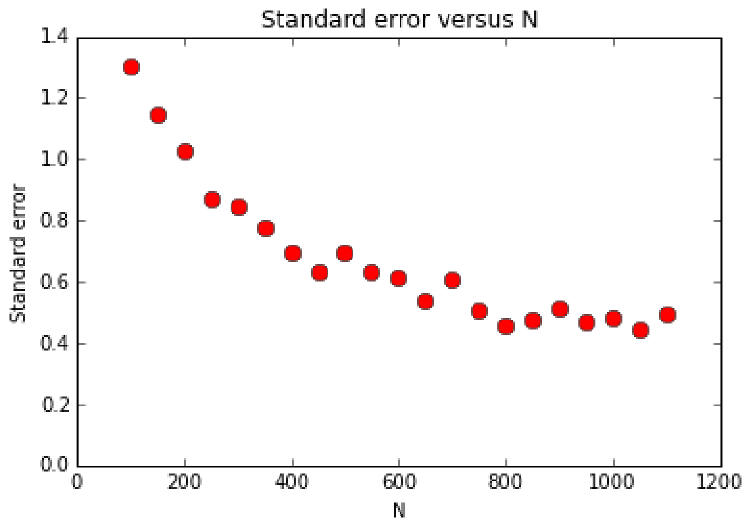

Figure 3: Standard error versus N

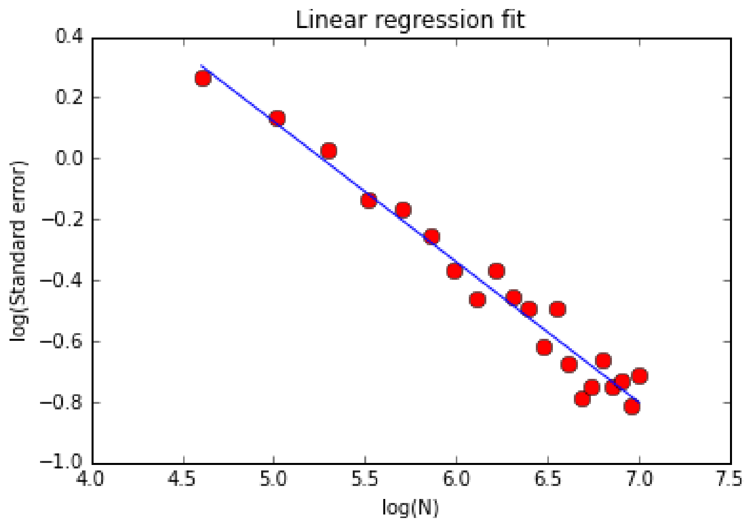

Figure 4: Standard error versus N

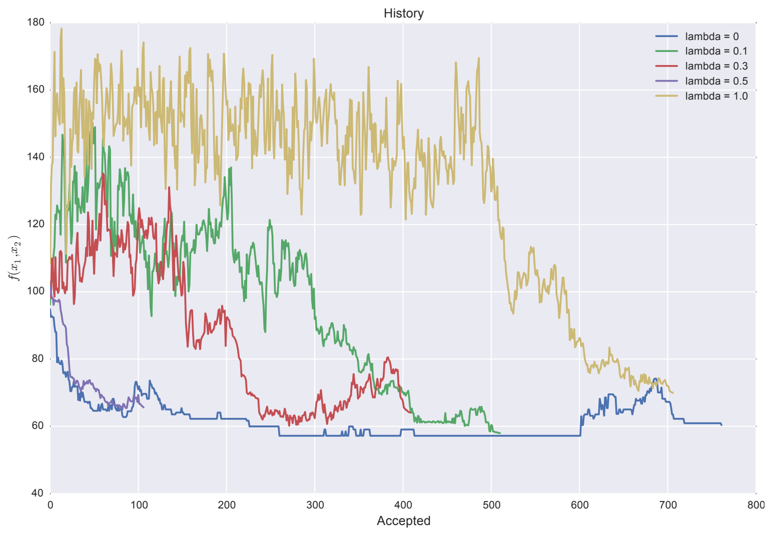

Figure 5: Simulated annealing with noise = $\lambda * random.uniform(0, 1)$ for different values of λ

7 Distributed Computing

One of the issues with scaling the problem from the 240 node mini graph of North Carolina to the full set of 24,244 nodes is that individual runs of the iterated cascade can run for many more iterations within a larger graph (particularly if the graph is well-connected), so each cascade may take much longer to complete. For this reason, we cap our cascade steps at 10. The value of this cap d also affects the scalability of number of starting nodes, which is described below. Because of this, a smaller value of d will make the independent cascade run faster since the whole tree is not explored thanks to the depth cap d.

Scaling d will have much less of an impact on a small graph than it will on a large graph because in a smaller graph, the independent cascade is more likely to terminate before completing d cascades. On a large graph, however, the independent cascade may run for many steps if the number of cascades is not capped. Furthermore, the distribution of the number of cascades that may run is likely to be right-tailed, so even if the independent cascades are parallelized, one particularly long cascade may take much longer than its neighbors to run, thereby mitigating some of the benefits of parallelizing the Influence function.

Let N be the umber of independent cascades run within calculations of the Influence function. Scaling N should linearly increase the amount of computing time for the Influence function, because increasing the number of simulations does not increase the computing time of each. As explained below, however, the Influence function lends itself well to parallelization, so increases in wall time can be minimized by parallelizing the Influence function. As noted above, however, there are diminishing returns in the accuracy of the Influence function by scaling N, so some intermediate value of N should give the optimal balance of low standard error and quick compute time.

Scaling the number of starting nodes will have different impacts on the Simulated Annealing and Greedy Algorithms. In the Greedy algorithm, the first iteration considers the Influence of each of the j nodes in the graph, and selects the node with the maximum influence. The second iteration considers the Influence by adding each of $(j-1)$ remaining nodes, and so on to $(j-k)$. For large graphs where k (the number of starting nodes) is much smaller than j (the total number of nodes in the graph), the difference between j and $(j-k)$ is negligible, and so incrementing k should lead to an essentially linear increase in computing time. That is, adding another starting node should add an amount of computation that is not much less than the amount of computation for a single node.

In the Simulated Annealing algorithm, however, changing the number of starting nodes will affect the independent cascade function time though there is no increase in the time we take to make node swaps for the next iteration or to run the simulated annealing itself. Hence the major bottle neck is the Independent cascade function for the Simulated annealing algorithm.

8 Parallelization Strategies

1. **Greedy Algorithm** - The Influence function lends itself very well to parallelization, because individual runs of the independent cascade are relatively fast and do not depend on one another. Also, with a cap on the number of cascade steps, they should each take roughly the same amount of time. Therefore, within both the Simulated Annealing algorithm and the Greedy algorithm, parallelizing the independent cascades within each run of the Influence function should lead to significant time improvements. To strategy to parallelize for the Greedy algorithm will be to split the graph nodes over the partitions

and independently run them over different cores since one computation does not affect the other.

2. **Simulated Annealing** - The algorithm is serial execution and the nodes on which the Simulated annealing operates in the i^{th} iteration is a function of the $(i-1)^{th}$ iteration. Hence the parallelization cannot be carried out on the nodes like the greedy algorithm. Instead we can parallelize the independent cascade function by parallelizing the N runs which the independent cascade runs. Since each of the runs is independent of a set of starting nodes, we can speed up Simulated annealing on the whole.