AC290 EXTREME COMPUTING: REPORT 2

Bleeding Edge

Gioia Dominedo, Christian Junge, Abhishek Malali and Isadora Nun

October 12, 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 nodes and the edges between them, and examining how the nodes within this graph activate one another. The nodes represent individual Yelp reviewers within the United States, the edges represent the connections between these reviewers, and activating a node represents that reviewer being influenced by one of its connections. The goal of this problem 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. In this way, we can select an initial set of reviewers to target with advertising and incentives that will have the maximum influence that spreads throughout the network of Yelp reviewers.

2 What is independent cascading?

In the independent cascading approach, one or several nodes begin the simulation as activated. The number of starting nodes is defined as k. In each iteration of the algorithm, each activated node has one chance to activate the nodes that it shares edges with. 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. For the purpose of limiting total computation time, there is also an optional parameter d, or depth, that limits the number of cascade iterations that are permitted to run.

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 is a method for finding a starting set of size k with a high number of activations without searching every possible set of starting nodes. In its first iteration, it finds the single node that has the highest value of the influence function when run over 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 k 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 set of nodes that is created by adding the next high-activation node to the size-n set may not be the best size(n+1) set possible.

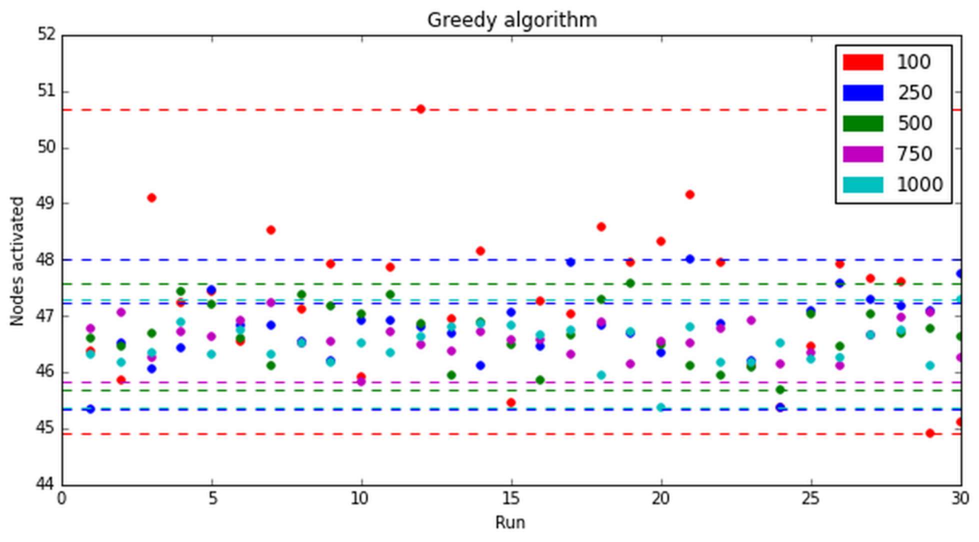

Figure 1: Plot of minimum and maximum activations for different number of runs

As can be seen in Figure 1, as the number of runs of the independent cascade within the greedy algorithm increases, the distance between the maximum and minimum activations decreases. The minimum bound also increases as expected with the increase in the number of runs for the greedy algorithm. This means that uncertainty due to stochasticity in the influence function can be reduced by increasing the value of N. There is a trade-off here, as increasing N increases the amount of computation, and displays diminishing returns.

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 randomly selected and the value of the influence function for N simulations is calculated. Then this starting set is perturbed by switching some of the starting nodes, and the expected value of this new set is calculated. The next iteration of the algorithm will use either the newly perturbed set or the set from the previous iteration, and it will select which of these two randomly with a probability that is related to the difference in "Energy" (in this case, how many more nodes one set activates versus the other set, denoted ΔE) and the "temperature" (T) of that given point in the algorithm. The temperature is a parameter that changes throughout the algorithm to adjust the probability of accepting new nodes. This probability is given by the equation $P(X = j) = \exp(\frac{-\Delta E}{T})$. If the temperature is high, the algorithm is more likely to accept a new set even though it returned a lower value of the influence function. Varying the temperature throughout the iterations of the algorithm allows it to consider sets of nodes that do not look promising at first glance, but may, with slight modification, yield better sets than those that were previously being considered.

The advantages of the simulated annealing algorithm are that it can search a very large sample space without considering an extremely large set of possibilities, and that it does so in a strategic way by iterating upon promising tracks, but not constraining itself to one small subset of the sample space.

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

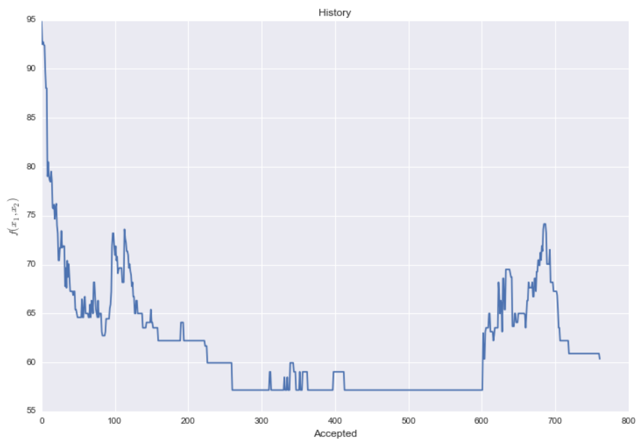

Figure 2: Simulated Annealing

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

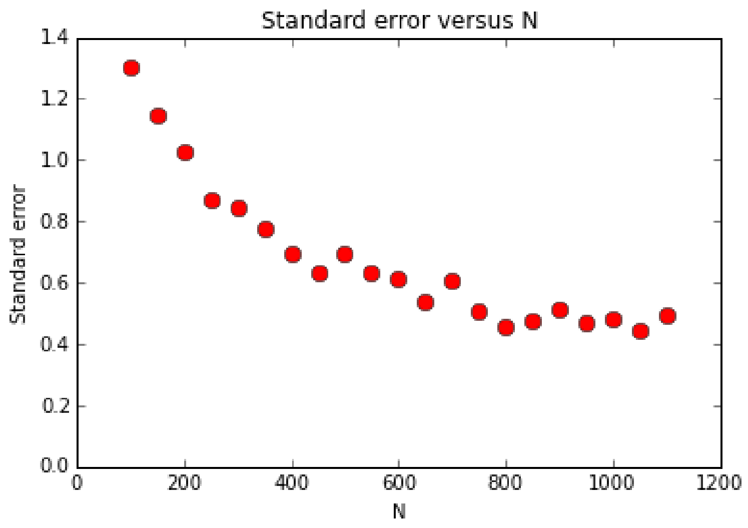

Figure 3: Standard error versus N

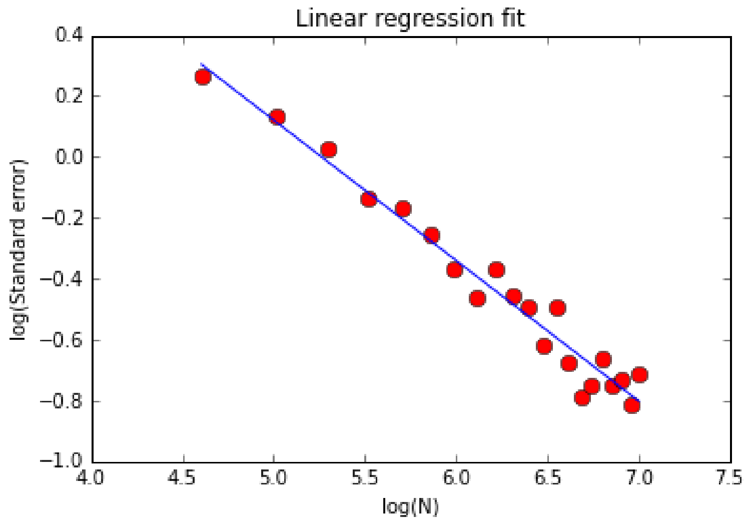

Figure 4: Standard error versus N

will yield a more accurate estimate of the average influence for a given set. This tightening of the standard error is proportional to $\frac{\alpha}{N^{\lambda}}$. By fitting the data with a linear regression we obtain $\frac{11.34}{N^{0.46}}$ for the mini-subset of the North Carolina graph. 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. The tradeoff between this uncertainty and compute time must be made carefully so that results are sufficiently accurate, but do not take too long to compute.

Also, as described above, there is uncertainty that either the Greedy Algorithm or the Simulated Annealing Algorithm will return the actual true optimal starting set with the maximum influence. This is because neither approach considers every possible set of starting nodes, so either approach may fail to find the true optimal set.

Within the Simulated Annealing Algorithm, high levels of uncertainty in the value of the influence function for each set of starting nodes considered cause the algorithm to fail to come to local minima. This is displayed conceptually in Figure 5 for the Traveling Salesman

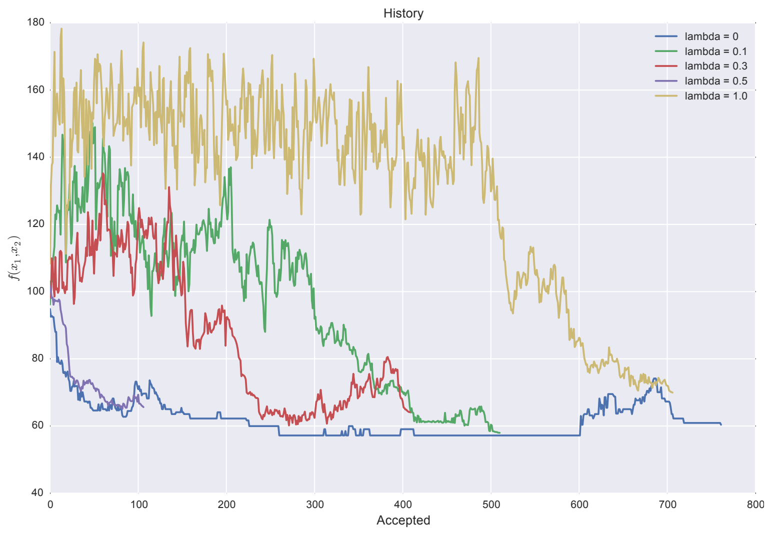

Figure 5: Simulated annealing with noise = $\lambda * random.uniform(0, 1)$ for different values of λ

problem, where varying levels of uncertainty have been introduced into the distances between cities. The ideal case is to be very certain of the distances so that the algorithm can converge properly on minimum distances. Analogously in the Influence Maximization Problem, if the value of N is made too low to speed up computing time, Influence function values will be too noisy for the Simulated Annealing algorithm to converge on maximum influence values.

7 Distributed Computing

For the purposes of this problem, we considered the problem at three scales: a mini subset of all of the reviewers in North Carolina (240 nodes), the full set of all the reviewers in North Carolina (24,244 nodes), and the full set of all of the reviewers in the United States (350,620 nodes). The purpose of breaking the set up in this way was to develop methods that could be run with a reasonable length of computing time, and then examine the issues related to scaling from a small subset to the entire set.

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 could potentially run for many more iterations within a larger graph (particularly if many of the nodes in the graph were connected through chains and in isolated pockets, rather than being well connected within the whole set), so each cascade could 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 number 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.