Simulated Annealing Algorithm for Graph Coloring

Prakhar Agrawal, Sverre Akersveen, Martin Hallén

May 25, 2016

1 Introduction

The goal of the project was to code and experiment the Markov Chain Monte Carlo (MCMC) method for the problem of graph coloring. One component of the project was to get familiar with the idea of simulated annealing, where the temperature parameter in the MCMC is lowered progressively. The main task of this project was to tune the parameters of the simulated annealing algorithm in order to optimize the cost of the graph coloring.

2 Method

2.1 Cooling schedule

Since there is no generically optimal cooling rate, we had to find the best way to reduce the temperature of the graph empirically (Carr). We had several different ideas on how this could have been done, and considered many different implementations. Among the rejected propositions, one can find schedules that changed the B-value on only a certain number of the time steps (for instance by introducing a modulo test on t) and a dozen different asymptotic functions for increasing the B-value. Below, we will present a handful of the more successful schedules. We also experimented by changing $B_0 (= 1/T_0$ where T_0 is the initial temperature) according to different functions.

We tried functions like $B = B_0 * t, B = B_0 * t^2, B = B_0 * exp(t), B = B_0 + 100 * t, B = 1/(1 - t/N_{steps})$ and $B = (1/(1 - t/N_{steps}))^2$, where B is the value of the function after t iterations and we checked the best value of the cost function it gives in N-steps(total number of iteration) and we found that our results improved if we increase N- steps as well as when we increase Q(the number of colors). Of the above functions $B = (1/(1 - t/N_{steps}))^2$ performed best on the basis of the cost. It is expected as we can observe from the graph of this function that it is increasing at moderate rate and also it is approaching ∞ which matches with our expectation. In this part of our study, it was important to experiment with cooling rates that were higher and lower than the provisionally optimal cooling rate.

3 Results

The results of the different cooling schedules can be studied in figure 1. Here, we are applying the simulated annealing algorithm with $C = 20$ and $Q = 3$ on an Erdos-Renyi graph with $|V| = 1000$. Among the functions we are comparing in this plot, one can observe that the ones where the cost decreases slowest to begin with gets the lowest cost when the algorithm terminates. It also indicates that the cooling schedule proposed above, $B = (1/(1 - t/N_{steps}))^2$, reduces the cost better than the other mentioned schedules.

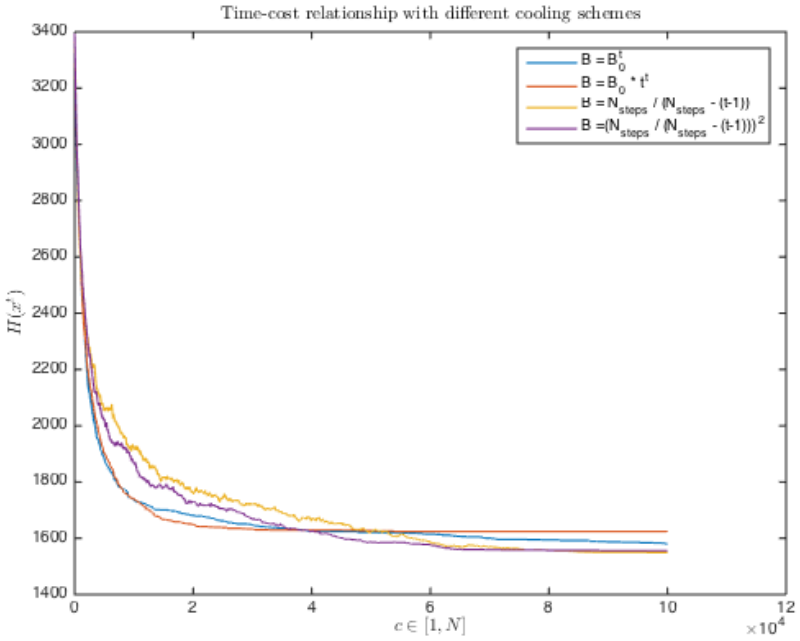

Figure 1: A plot of $H_{min}(x^t)$ for different cooling schemes.

3.1 H_{min} as a function of Q and C

Given a fixed number of colors(3, 4, 5, 6 or 7), there is a (close to) linear relationship between C and H_{min} as proposed by figure 2. Thus, the number of conflicts increases as the graph gets denser. Also, by reducing the number of allowed colors, we increase the number of conflicts. Both these findings follows intuitively from the problem description.

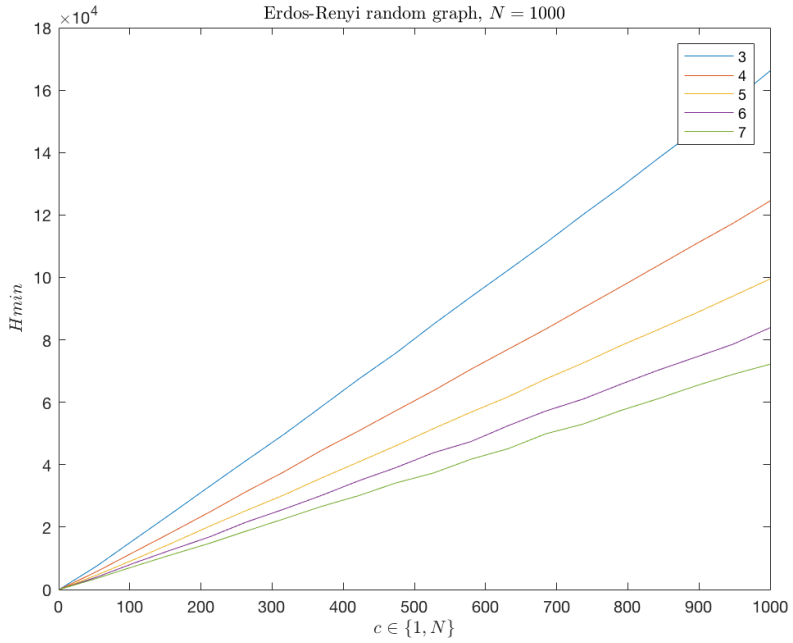

Figure 2: A function of H_{min} for different values of $\|Q\|$.

4 Conclusion

During this project, we have experienced how to optimize the Simulated Annealing Algorithm through tuning of the parameters of the cooling schedule. We have seen how reducing the temperature too quickly may result in getting stuck in a local minimum and how reducing the temperature too slowly will not yield the optimal result within the provided time frame. We have seen through our study how the best cooling schedule seems to be the one where we increase the inverse of the temperature, B, at a rate of $B = (1/(1 - t/N_{steps}))^2$.

5 References

Carr, Roger. "Simulated Annealing." From MathWorld–A Wolfram Web Resource, created by Eric W. Weisstein. Retrieved from mathworld.wolfram.com/SimulatedAnnealing.html