Simulated annealing algorithm for graph coloring

Xavier Fontaine, Thomas Grivaz, Antoine Mougeot

May 25, 2016

Abstract

This report summarizes our methodology and results in the context of experimenting the Markov Chain Monte Carlo method for the problem of graph coloring.

1 Introduction

TODO

2 Metropolis chain

In this section we run experiments with β fixed and we apply the Metropolis-Hastings algorithm as described in the project statement. We analyze the influence of the initial value of β on the minimal value of the Hamiltonian reached by our algorithm. We can think of β as the inverse of a temperature and therefore we note $T = \dfrac{1}{\beta}$.

In these experiments we fix an Erdös-Renyi graph G with parameters $N = 100$ and $c = 20$. We use $d = 5$ colors. We test three different values of temperature: $T = 10$, $T = 1$ and $T = 0.1$ and we run the simulations with 3000 iterations. Here are the curves we obtain:

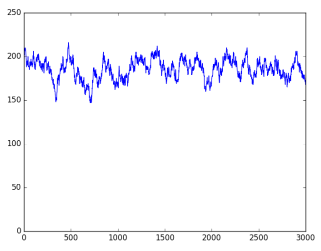


Figure 1: $T = 10.0$ and $\min H = 147$

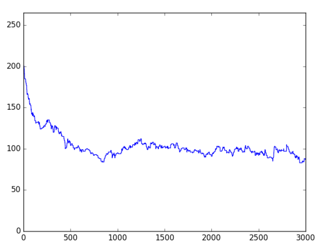


Figure 2: $T = 1.0$ and $\min H = 83$

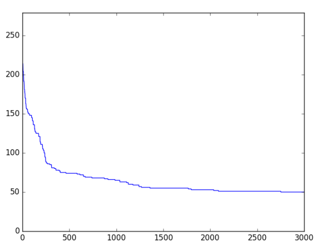


Figure 3: $T = 0.1$ and $\min H = 50$

- When T is high (i.e. β is small), the probability to accept a move that increases the Hamiltonian is high (for $T = 10.0$ it is roughly 80%) and consequently the values of H vary a lot. Therefore we observe a lot of fluctuations and the chain does not converge into a precise state. Moreover, the final Hamiltonian value is not equal to the minimal Hamiltonian value. We observe also that since β is small, π_β is far away from π_∞ and then the minimal Hamiltonian value that we reach is really bigger than the global minimum we could reach. With the graph we used and these settings we obtain an average minimal value of 153 for H.

- With an intermediate value of T ($T = 1.0$ here) the previous phenomenon is less visible. The probability to accept a move increasing the Hamiltonian is indeed smaller (about 10% now) and therefore we observe a global decrease of H. However we can still see some fluctuations but the final Hamiltonian value is near from the minimal Hamiltonian value reached during the 3000 iterations. The distribution π_β is still far from π_∞ and therefore the minimal Hamiltonian value we obtain here (its average is 80) is still much bigger than the global minimum.

- When T is low, the probability to increase H is very low and therefore we obtain a function $t \mapsto H(x^t)$ that is non-increasing. Now the distribution π_β is closer to π_∞ but the probability to get stuck in a local minimum is really high. Therefore the final Hamiltonian value is fast always equal to the minimal Hamiltonian value obtain during the iterations because the chain does not manage to leave this local minimum. However we obtain smaller values for H ($H = 50$ in our case on average).

Therefore we can conclude that when T is fixed we have to choose a low value for T in order to obtain small values for the Hamiltonian. The drawback of this method is that we are likely to get stuck in a local minimum. However if we increase T we will need too many iterations in order to reach small values of H.

In order to obtain better results we will use the method of *Simulated Annealing* which decreases the value of T during the iterations.

3 Simulated Annealing

In this section we will describe how we chose and tuned the parameters of our algorithm. In order to minimize the bias we could have regarding the graph we're trying to color, several graphs were considered for each value of a parameter: one random graph with $N = 100$, $c = 5$ and $q = 3$ (which was the graph provided on the project webpage, that we call G_1), one random graph with $N = 200$, $c = 40$, $q = 7$ (G_2), one random graph with $N = 50$, $c = 10$, $q = 3$ (G_3) and finally a random graph with $N = 100$, $c = 30$, $q = 7$ (G_4). Besides G_1, we chose graphs with high edge probability that result in high final energy after having run the Metropolis algorithm so that's it's easier to notice differences between choices of parameters.

3.1 Initial Temperature

According to Kirkpatrick [1], a suitable temperature T_0 is one that results in an average increase of acceptance probability p_0 of about 0.8. The value of T_0 depends on the scaling of our cost function and hence is problem specific. To estimate this, we conducted an initial search on each graph where all increases are accepted and calculated the average increase over a fixed number of iterations. The initial temperature is given by :

$$T_0 = -\frac{\mathbb{E}(\Delta_+)}{\ln(p_0)}$$

Where $\Delta_+ = H(x^{new}) - H(x^0)$ is a strictly positive increase. We also tried different base acceptance probabilities (0.5, 0.3) and several fixed values of T_0, but this method gave us the best overall results.

3.2 Cooling Function

Several ways of decreasing the temperature were considered. We found three cooling functions that gave good results:

- Exponential schedule:
$$T(t) = T_0\alpha^t$$
We considered $\alpha \in [0.7, 0.95]$ and we selected the best value of α with running experiments with a step of 0.01.
- Linear schedule:
$$T(t) = T_0 - \eta t$$
with $\eta \in [0.05, 0.4]$ and a step of 0.05.
- Power schedule:
$$\forall t \geqslant 1,\ T(t) = \frac{T_0}{t^\gamma}$$
with $\gamma \in [0, 1]$, and a step of 0.1.

Moreover since we did not want to have too low temperatures we chose a minimal value for T:

$$T(t) = \max(T_{\min}, T(t)).$$

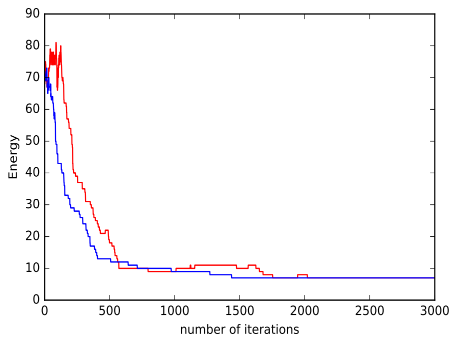
We found that $T_{\min} = 0.1$ is a good value.

In order to find the best values for the parameters α, γ and η we ran the Metropolis algorithm on our four graphs with different values for those parameters and we averaged the final energy obtained. We then kept the value for which the final energy was the minimum and ran the whole process again several times to make sure that it was indeed the best overall parameter. Note that even though differences in results were very small (the differences in energy between parameters were in the 5% range), we noticed that the optimal parameter was consistent from one run to another. We checked also that the parameters we obtained were optimal for each graph taken separately.

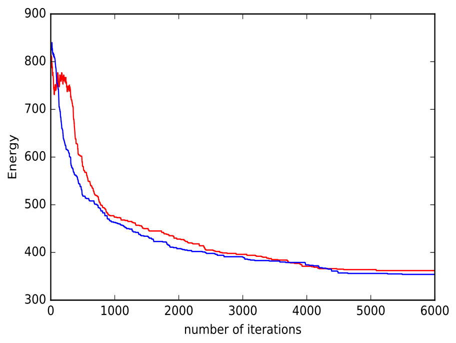
Finally we found the following values for our parameters. These are the values that gave the best average results after several runs of the Metropolis-Hastings algorithm.

α	η	γ
0.85	0.25	0.5

We compared the three schemes with these parameters by plotting the evolution of $H(x^t)$ for all schemes:



(a) G_1



(b) G_2

Figure 4: Evolution of the energy of the Metropolis algorithm for two graphs. The red curve represents the evolution of the energy using a linear scheme while the blue curve represents the evolution of the energy using an exponential scheme.

As we can see from figure 4, results are pretty similar. The exponential schedule tends to decrease faster but obtains better results overall, as such, we chose this schedule to decrease the temperature.

3.3 Epoch length

We found out that decreasing the temperature at each iteration was not the best cooling scheme. Indeed when T is low, the probability to accept increasing moves is small and therefore we need to stay at the same value of the temperature before accepting some moves. If we decrease directly the value of T we do not really take profit of the effect of intermediate temperature values. That is why we tried to decrease the temperature every x iterations, where $x \geqslant 2$ is an integer to find. In order to find this epoch length we run the Metropolis algorithm several times with different values of x and we chose the value of x that gave the best average results. We also set the number of iterations to $10,000$ to make sure to reach convergence.We summarize our results in the following tabular:

Epoch	1	2	3	5	10
Average minimum Hamiltonian	131.17	126.77	126.98	124.67	127.59

Table 1: Epoch length results

We find that an epoch of 5 gives good results. However, as we said it, for low temperatures the acceptance probability of moves that increase the Hamiltonian is low and therefore we have to stay at the same temperature during few iterations to see the effect of this temperature value. This remark suggests to try non-constant epoch lengths, i.e. to wait longer times before decreasing the temperature when T is low.

A possible scheme to implement this is the following:

$$\begin{cases} \text{if } T > 5, & \text{decrease temperature every 5 iterations} \\ \text{if } T > 1, & \text{decrease temperature every 10 iterations} \\ \text{if } T > 0.1, & \text{decrease temperature every 20 iterations} \end{cases}$$

Even though differences are really small, an epoch length of 5 seems to be the most suited for our problem. Surprisingly enough, we observed that waiting a certain amount of iterations at the beginning of the algorithm before decreasing the temperature was worse than decreasing straight away

a verifier

References

[1] S. Kirkpatrick. Optimization by simulated annealing - quantitative studies. *J. Stat. Phys.*, 34:975–986, 1984.