Markov Chains - Mini Project
Application of Monte-Carlo methods to community detection

Nina Mainusch, Alec Flowers and Hilda Abigél Horváth

May 6, 2022

Abstract

Our goal is to apply the variants of Metropolis-Hastings algorithm to the community detection problem. We saw that the mixed Houdayer-Metropolis algorithm could converge faster to the optimal solution with fewer iterations than vanilla Metropolis, but that the vanilla Metropolis algorithm performed very well and is computationally efficient. We also looked at the limiting performance of the algorithms by increasing the difficulty of the problem setup and saw similar behavior for all three near the theoretical critical value.

1 Introduction

In this project we used the **Stochastic Block Model** to model the network of connections. For simplicity, we assumed that there are only two communities in the observation graph.

We generate this G graph in the following way: G has N nodes, denoting the members of the network. Each node belongs exclusively to a community, $x_i^* \in \{\pm 1\}$, which is chosen uniformly at random. The edges between nodes symbolize the connection between the members with the assumption that members of the same community are more likely to be connected to each other than to members of other communities. Let e_{ij} denote the bidirectional connection of node i and j, thus $e_{ij} = e_{ji}$. If $e_{ij} = 1$, then there is an edge between the two nodes, otherwise there is no edge. If a, b are positive constants, we can generate the edges independently assuming

$$\mathbb{P}[e_{ij} = 1 | x_i^* x_j^* = 1] = 1 - \mathbb{P}[e_{ij} = 0 | x_i^* x_j^* = 1] = \frac{a}{N}$$
$$\mathbb{P}[e_{ij} = 1 | x_i^* x_j^* = -1] = 1 - \mathbb{P}[e_{ij} = 0 | x_i^* x_j^* = -1] = \frac{b}{N},$$

where $i < j$ due to the symmetry of the edges.

Our goal is to detect the communities from graphs generated as described above, where the x^* are unobserved. We only consider values of a, b such that $(a-b)^2 \leq 2(a+b)$, since it can be shown that otherwise the community membership cannot be detected.

To assess our estimation of the allocation of community memberships, we use the following *overlap* quantity, where $\hat{x}_i$ is our membership estimate of node i:

$$q_N = \frac{1}{N} |\sum_{i=1}^{N} x_i^* \hat{x}_i|.$$

2 Algorithms

2.1 Metropolis algorithm

We begin with describing our implementation of the standard Metropolis algorithm. The initial random estimate of the community memberships is denoted by $\hat{x}$. In each step we choose a random node and decide whether to flip its membership to the other community by calculating the energy of the current state and the new state as

$$H_{state} = -\sum_{1 \leq i < j \leq N} h_{ij} x_i x_j, \quad \text{where} \quad \frac{1}{2}\left(e_{ij} \ln \frac{a}{b} + (1 - e_{ij}) \ln \frac{1 - \frac{a}{N}}{1 - \frac{b}{N}} \right).$$

We accept the change with probability $\exp(-\beta \cdot (H_{new} - H_{current}))$. We iteratively choose a new random node and repeat this procedure, which we call the Metropolis step.

2.2 Houdayer algorithm

Here instead of having one chain, we let two chains run in parallel, denoted by $x^{(1)}$ and $x^{(2)}$. In a Houdayer step we calculate the local overlap $y_i = x_i^{(1)} \cdot x_i^{(2)}$, which yields clusters of ± 1, depending on whether the cluster assignments agree or disagree in the two chains. We then randomly choose a node with value -1. Afterwards, we flip the membership assignment in both chains of all the nodes in the -1 cluster associated with the chosen node. Then we perform a Metropolis step and repeat the two steps iteratively.

(Theory) Is the chain on the pair configurations constructed from the Houdayer move ergodic?

The chain on the pair configurations constructed from the Houdayer move is not ergodic. The state space of such a chain is $S = \{0,1\}^n \times \{0,1\}^n$. Here we construct a simple example of a state in the chain which is recurrent and therefore not ergodic. We construct a chain with state space $S = \{0,1\}^2 \times \{0,1\}^2$ that is $[(-1,-1),(-1,-1)]$. The Houdayer algorithm calculates the local overlap and chooses randomly a node with -1 and flips the cluster assignment. In our case there is no node with -1 and therefore no flip occurs. We actually can never leave this state. Note that even offset nodes in such a setup, for example $[(1,-1),(-1,1)]$, will only flip back and forth between two states. This counter-example shows that the chain on the pair configurations is not ergodic.

2.3 Mixed Metropolis-Houdayer algorithm

This version of the Houdayer algorithm differs from the original one in regard to the frequency of the Houdayer steps. We do not perform them after every Metropolis step, i.e. every 2 steps, but every $n_0 > 2$ steps.

3 Convergence time

In order to investigate the convergence time of the various algorithms, we ran simulations on graphs of $N = 100$ nodes with $b/a = 0.25$ and node degree $d = 12.5$. For each algorithm we performed 100 simulations and recorded the first step with perfect overlap $q_N = 1$.

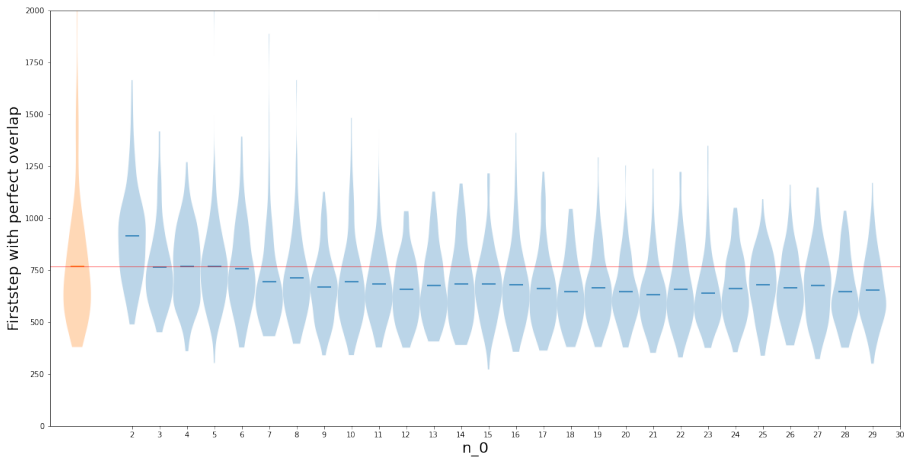
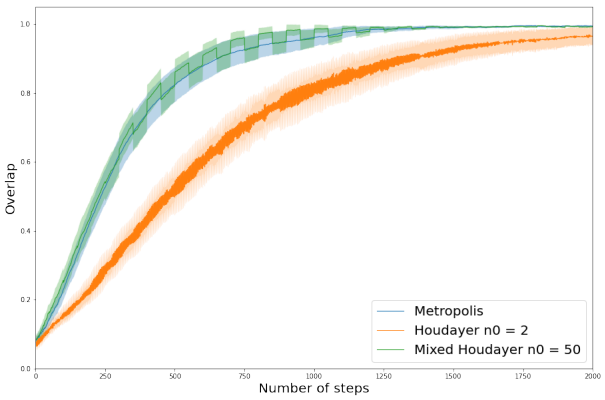


Figure 1: Convergence time of simulations of the different algorithms. The orange violin plot denotes the Metropolis algorithm, the blue ones denote the Houdayer and Mixed-Houdayer algorithm for n_0 varying on the x-axis.

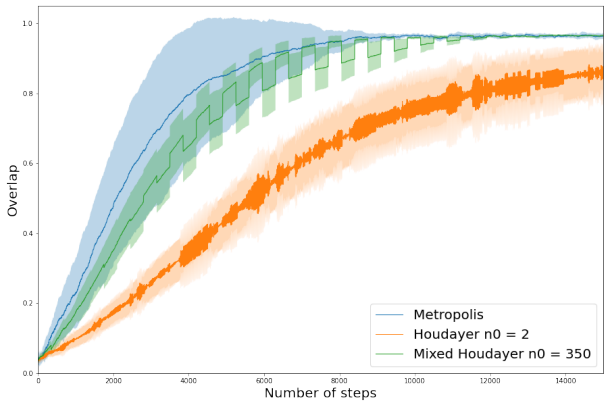
We display our results in Figure 1. Each violin plot shows the distribution of the first step of reaching the perfect overlap over 100 simulations for the Metropolis (colored orange) Houdayer and Mixed-Houdayer algorithms (colored blue), where the frequency of the Houdayer step n_0 ranges from 2 to 30. The mean convergence time is marked by the thicker line in the middle of each violin plot.

As it can be seen, having Houdayer steps improves the mean convergence time for $n_0 > 7$ in this graph configuration. In theory, the advantage of cluster moves as in the Houdayer algorithm begins to make a difference if N is large. Unexpectedly, in the setup of 100 nodes we could already capture improvements in convergence time, such that this number of nodes was big enough to observe the strength of Houdayer.

Next we increased the number of nodes to $N = 500$ to see if this phenomenon holds for a larger graph. The reason being we believed the overwhelming might of Houdayer would be unleashed as N grew large. Because of running time constraints we chose to just run the metropolis algorithm vs. Houdayer with $n0 = [2, 350]$ for 50 iterations.



(a) $N = 100$ $\frac{b}{a} = .25$, 100 simulations



(b) $N = 500$ $\frac{b}{a} = .25$, 50 simulations

Figure 2: Average empirical overlap of different algorithms as a function of time.

Our expectation is that the local flips from Houdayer will allow for a faster convergence because it can move out of local minimums more quickly. In reality, we experienced that for bigger n_0s the Mixed-Houdayer is the one who can asymptotically beat the simple Metropolis algorithm, which is observable in Figure 2b as the Mixed-Houdayer algorithm for $n_0 = 350$ rises above the Metropolis algorithm as the number of steps grows large. As it can be seen both in Figure 2a and Figure 2b, the Houdayer with $n_0 = 2$ can not compete with the Metropolis in neither of the cases $N = 100$ nor $N = 500$.

4 Phase Transitions

4.1 Theoretical phase transition: critical value r_c

Taking as parameters the average degree of the graph $d = \frac{1}{2}(a+b)$ and $r = \frac{b}{a} \in (0, 1]$ we can solve for the critical value r_c (as a function of d) at which the phase transition occurs:

$$
\begin{aligned}
(a-b)^2 &\leq 2(a+b) \\
(a-b)^2 &\leq 4d \\
(a-b) &\leq 2\sqrt{d} \\
\frac{(a-b)}{2} &\leq \sqrt{d} \\
\frac{(a+b)}{2} - \frac{(a+b)}{2} + \frac{(a-b)}{2} &\leq \sqrt{d} \qquad (1)
\end{aligned}
$$

Where we add and subtract by d in (1). From (1) we can solve for both a and b.

$$
\begin{aligned}
d + \frac{a - a - 2b}{2} &\leq \sqrt{d} \\
d - b &\leq \sqrt{d} \\
b &\geq d - \sqrt{d}
\end{aligned}
$$

$$
\begin{aligned}
-d + \frac{b - b + 2a}{2} &\leq \sqrt{d} \\
-d + a &\leq \sqrt{d} \\
a &\leq d + \sqrt{d}
\end{aligned}
$$

Therefore $\frac{b}{a} \geq \frac{d-\sqrt{d}}{d+\sqrt{d}}$ which is $\in (0, 1]$ and r_c is the critical value $\frac{b}{a} = \frac{d-\sqrt{d}}{d+\sqrt{d}}$. As the node degree d becomes large, this critical value r_c approaches 1. This means that for more and more connected graphs, a and b can get closer and closer together and the Metropolis algorithm will still be able to detect the communities.

4.2 Empirical phase transitions

We can use the theoretical computations to guide our empirical tests to see if we can replicate the expected behavior around the phase transitions critical value. We have compute constraints and therefore can only run these algorithms for a finite number of steps on a reasonably sized graph which is why we may have gaps between our results and the theory.

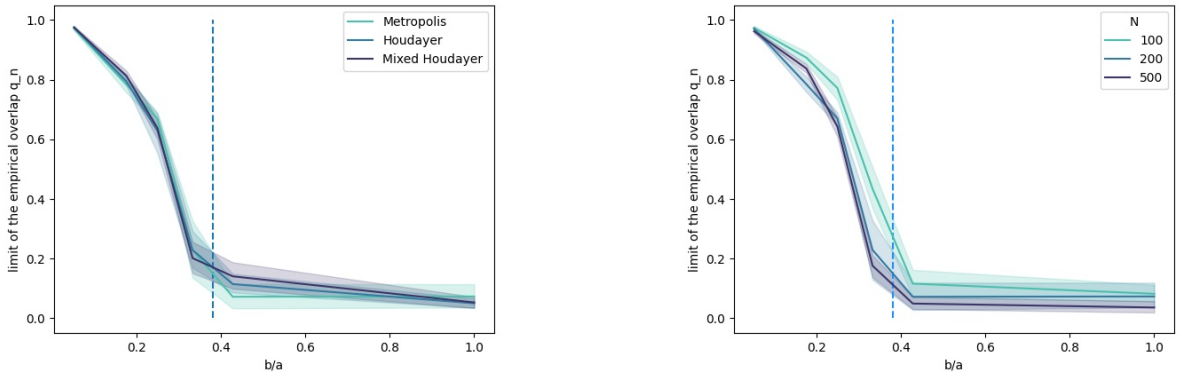
4.2.1 Limiting performance of different algorithms

To look at the limiting performance of the three algorithms we ran them on a graph of $N = 200$ with node degree 5. To get $lim_{t\to+\infty} q_N(t)$, each algorithm was run ten times for $N^2 * \log(N) \approx 90,000$ iterations and for several ratios of a and b. The parameter n_0 for the Mixed-Houdayer algorithm was set to 9.

To plot our results we display the ratio of a and b against the limiting empirical estimation overlap $lim_{t\to+\infty} q_N(t)$. The corresponding theoretical phase transition is $r_c = 0.382$ which we indicate by a dashed vertical line.

We expect to see that all algorithms are effected by the phase transition, but that the Houdayer algorithm can explore the space better than the Metropolis algorithm, such that the phase transition will be steeper for Houdayer than for Metropolis. The mixed algorithm will most likely exhibit a trade-off between these two.

Inspecting the results in Figure 3a, for ratios approaching and beyond the theoretical phase transition point the algorithms cannot converge anymore and $lim_{t\to+\infty} q_N(t)$ drops to 0. Different to what we expected, the empirical means and 95% confidence intervals around it overlap for each of the three algorithms. Since there is no observable difference in the convergence behavior of these three algorithms for the chosen node degree and N, we conclude that all algorithms perform identically.



(a) Limiting performance

(b) Pinpoint phase transition

Figure 3: The dashed lines correspond to the theoretical phase transition point $r_c = 0.382$. (a) Performance plot of the different algorithms for $N = 200$, node degree 5 and 90,000 iterations. (b) Phase transition plot of the Metropolis algorithm for $N \in \{100, 200, 500\}$ and node degree $d = 5$.

4.2.2 Pinpoint the phase transition

To pinpoint the phase transition, we decided to focus on the Metropolis algorithm. N ranges between $\{100, 200, 500\}$ and the algorithm runs for ten times for $N^2 * \log(N)$ iterations each. We expect to see a steeper phase transition for larger N, close to the calculated theoretical transition point. In Figure 3b we can observe that the phase transition gets indeed steeper for larger N of 200 and 500 compared to the line for $N = 100$. Because we are operating on a finite graph with finite iterations we see our empirical overlap in Figure 3a and 3b drop slightly before the theoretical limit.