Jacob Libsman - 661723966 PHYS 4810

Final Project

For this project I'm going to reexamine our discussion on the Ising model and the algorithm we used to find solutions to the problem. In class, we discussed one specific application, which was the alignment of ferromagnetic material. To simplify the problem, I ignored external interferences such as a magnetic field, and assumed that the only property contributing to the total energy of the system was the interactions between adjacent particles without periodic boundary conditions.

Let each particle have magnetism $\sigma_i = 1$ or $\sigma_i = -1$, then the energy contributed to the total system for two particles is

$$-J\sigma_i\sigma_j$$

where J is a coupling constant. Therefore the total energy, or the Ising Hamiltonian, is

$$\mathcal{H}(\sigma) = -\sum_{i<j} J\sigma_i\sigma_j$$

Now lets take a look at the Maximum-Cut problem for graphs. The problem is almost identical in formulation, given a graph with nodes V and edges E, we would like to find two disjoint subsets V_1 and V_2 such that the sum of the weights of the edges between these two sets is maximized. In other words, assign every node to group $\sigma_i = 1$ or $\sigma_i = -1$, then add the weights w_{ij} between nodes if they are in different sets. This can be expressed with the following formula

$$\sum_{i<j} w_{ij}\frac{1-\sigma_i\sigma_j}{2} \tag{1}$$

manipulating this formula, we find that it is equal to

$$\frac{1}{2}\sum_{i<j} w_{ij} - \frac{1}{2}\mathcal{H}(\sigma)$$

So finding the Max-Cut of a graph is the same as the Ising Problem down to a factor of $-\frac{1}{2}$.

Max-Cut is known to be NP-Hard, so there is no known polynomial time algorithm to maximize/minimize this function. The algorithm we looked at in class to approximate a solution was the Metropolis algorithm.

The Metropolis algorithm is mostly simple. Chose one node or particle to swap the orientation of, then accept the change if we lowered the energy, and if not, accept the change with some probability. The other algorithm I would like to discuss here is the Goemans-Williamson algorithm for approximating Max-Cut.

Goemans-Williamson: Equation (1) can be altered with one simple change to describe a related problem such that maximizing the function is no longer intractable. Simply let each σ_i be a n dimensional vector where n is the number of particles/nodes. This makes the maximization problem fall under the category of semi-definite programming. Without going into too much detail, this involves maximizing a convex function over the intersection of a cone and a set of positive semidefinite matrices. I'll delegate maximizing this function to a solver made by experts at optimization problems, more on that later.

Now to choose which set each s_i should fall in, the trick is to pick a random hyperplane to separate all the vectors. If s_i lies above the random hyperplane, assign it the value 1, and otherwise the value -1. The specific details of the proof can be found in the original paper, but in short, the cut made from these random choices will lead us to a cut-value that is expected to be around 87% the value of the maximum cut. This value is the gold standard for approximation algorithms that solve Max-Cut, and is commonly used as a benchmark when evaluating new algorithms.

Next lets go over the ups and downsides that I found for both algorithms.

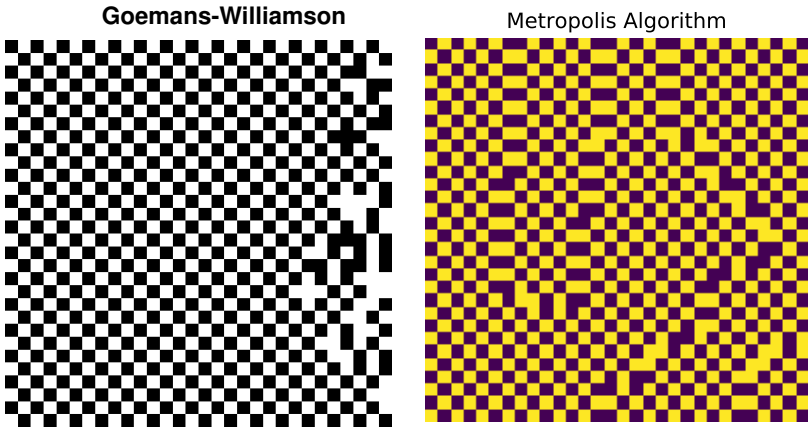
Starting with the Metropolis algorithm, the main difficulty for all applications of this algorithm will be to decide what temperatures to chose and what the tolerance for stopping is. There is a tradeoff between picking a wide range of temperatures and number of iterations with the runtime of your algorithm. The main computation is computing the change in energy after a particle is flipped. However, in the ising model, with a nice defined lattice of points, computing the energy is near trivial. Therefore (spoilers) the Metropolis algorithm will be the clear winner for solving the 2d-Ising problem. In my implementation, updating the energy only takes 4 $n \times n$ matrix sums, and a pointwise multiplication between two $n \times n$ matrices, where n^2 is the number of particles. Therefore, I was able to churn through 30 temperatures, with 1000 iterations for each temperature, for 100×100 matrices in a relatively short period of time. The only hard task is picking temperatures that will produce good results.

The Goemans-Williamson algorithm gave me quite a bit of trouble, mainly in selecting a semidefinite program solver. I ended up selecting a solver in MATLAB because the Python solvers were too convoluted to install on windows. And the problem with the MATLAB solver is the limitations of the size of the matrix I can use. The largest set of points I could use without running out of ram was somewhere between 30×30 points and 50×50 points, and the solver is extremely slow for matrices that large. The easy part of the algorithm is randomly sampling the vectors. In my MATLAB code, you can see that I take 1000 random trials since they are computationally inexpensive. This reduces the variance in taking one random selection to get as close to the maximum as we can get. The only upside was that the Goemans-Williamson had better optimal values for the minimum energy of the system for larger values of n.

For a set of 30×30 particles with $J = -1$ (weights are positive), the minimum energy calculated by my Metropolis algorithm will fall somewhere around -760, where the actual minimum is -870, which is about 87% of the minimum (interestingly, pretty much the expected value of the Goemans-Williamson algorithm). I used a range of 30 evenly spaced temperatures between 5 and 0.01, with 1000 iterations of flipping particles for each temperature and will take about 1.5 seconds to run.

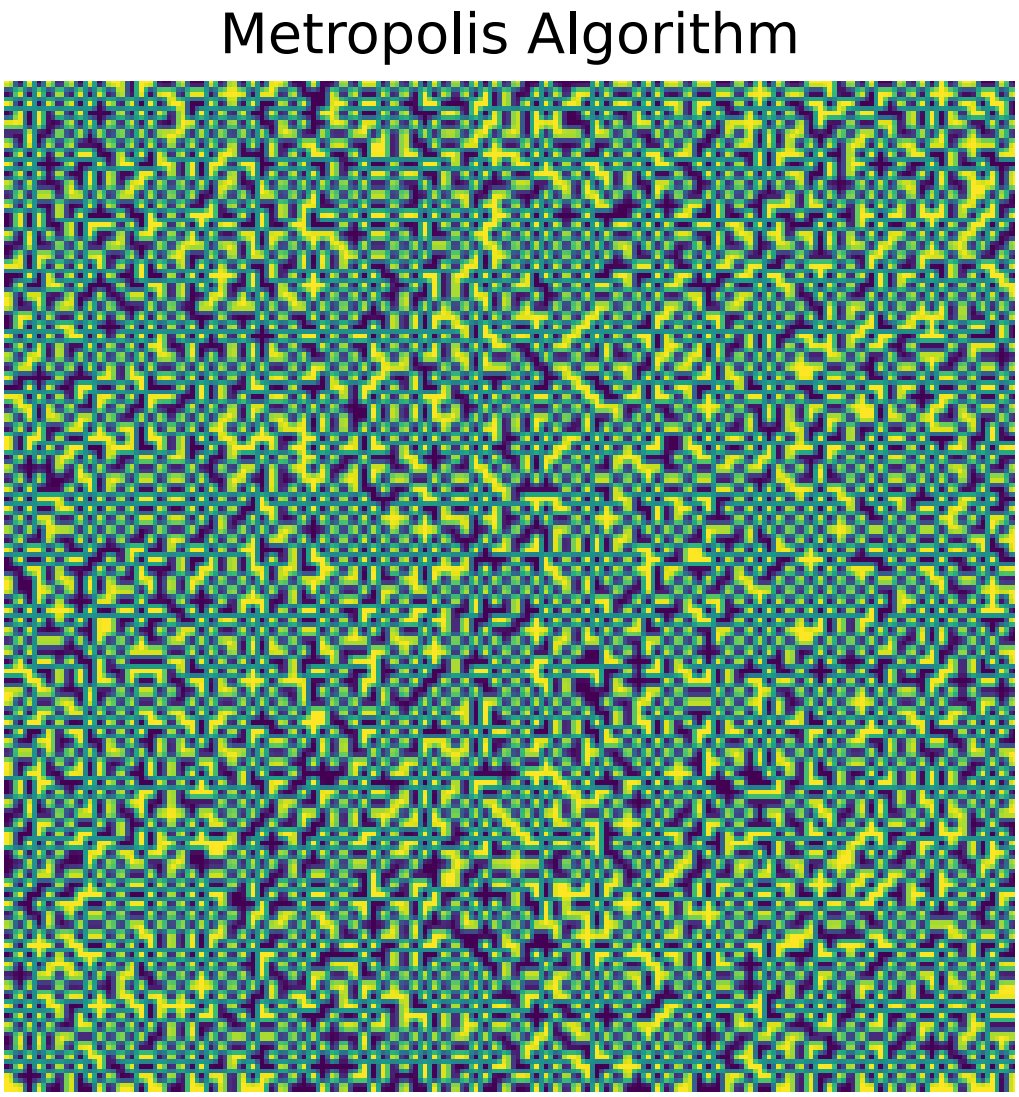
The minimum energy calculated on the same set of particles by the Goemans-Williamson algorithm was about -837, which is 96% of the minimum value! However, the time to solve the semidefinite program usually took around 14 seconds by itself. The rest of the computation didn't add much time at all, maybe a second at worst.

The plots of the final configuration are also interesting, Almost always, the Goemans-Williamson solution will produce that smudge of points you see on the right on one of the sides of the particles, where as the Metropolis algorithm has continuous segments where there is a defined and recognizable pattern, but isn't optimal.



Now, lets talk about why I think the Metropolis algorithm is a clear winner here. Its mainly due to the formula to update the energy after a particle has been flipped. After implementing a fast function to regulate this task, the computation time went way down, but coming up with this function took some time. For other applications related to magnetic interactions or Max-Cut, you will need to come up with a new function to update the energy for every application. This takes much more time then letting the Goemans-Williamson algorithm handle all the hard work for you. For the Goemans-Williamson algorithm, all you have to do is figure out how to represent your interactions with an adjacency matrix, and figure out the best number of trials for randomly sampling, which are easy enough tasks. And say we use the naive approach for updating the energy for the Metropolis algorithm and manually check every particle and its interaction with connected particles. This *greatly* increases computation time, to the point where there's no point running your algorithm on anything with meaningful size.

Also the Ising model doesn't have great guarantees on its accuracy. Using the same conditions as used with the 30×30 matrix, I used my Metropolis algorithm to approximate the minimum energy of a 100×100 matrix. The value computed was -5536, which is 56% of the minimum energy, which is -9900. Not great but not terrible, and the computation time was still low, only 3.3 seconds.



References:

Rhone, Trevor. Computational Physics, Rensselaer Polytechnic Institute, Troy, NY, 2021.

Haribara, Y., Utsunomiya, S., & Yamamoto, Y. (2016). A Coherent Ising Machine for MAX-CUT Problems: Performance Evaluation against Semidefinite Programming and Simulated Annealing. *Principles and Methods of Quantum Information Technologies,* 251–262. https://doi.org/10.1007/978-4-431-55756-2_12

Goemans, M. X., & Williamson, D. P. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM, 42*(6), 1115–1145. https://doi.org/10.1145/227683.227684

Grant, M., & Boyd, S. (n.d.). *CVX: Matlab Software for Disciplined Convex Programming.* CVX Research, Inc. http://cvxr.com/cvx/.