Week Seven PHY-480

Lewis

October 9, 2025

1. Ground state using a greedy strategy

Start with a random set of spins ($S_i = \pm 1$). Go through each spin one at a time and flip it only if doing so lowers the total energy. Repeat this process until no single flip can reduce the energy any further. The result is a locally minimal energy state.

2. Ground state using simulated annealing

Start with a random spin configuration and a high temperature then randomly flip spins and calculate the change in energy ΔE.

- If $\Delta E < 0$, accept the flip.
- If $\Delta E > 0$, accept it with probability $e^{-\Delta E/T}$.

Lower the temperature so the system becomes less likely to accept higher energy states and when the temperature is near zero, the system settles near the ground state.

3. Simulated annealing psedocode

1. Start with a random spin configuration.
2. Set an initial high temperature.
3. Repeat:
   (a) Pick a random spin and flip it.
   (b) Calculate the energy change.
   (c) Accept the flip if it lowers energy, or sometimes accept it if it raises energy (based on temperature).
   (d) Slowly decrease the temperature.
4. Stop when the temperature is near zero or energy stops changing.
5. The final configuration is the approximate ground state.

4. Showing that at $T = 0$, Metropolis MC is greedy

In the Metropolis Monte Carlo method, flips that raise energy are accepted with probability $e^{-\Delta E/T}$. When $T = 0$, this probability becomes zero, meaning only energy-lowering flips are accepted.S o $T = 0$, the algorithm behaves as a greedy algorithm.

Algorithms

1. **Prim's greedy algorithm:** Start from any vertex and repeatedly add the smallest edge that connects a new vertex to the growing tree. Continue until all vertices are included. Each step expands the connected portion of the graph using the minimum available edge.

2. **Dijkstra's greedy algorithm:** Begins at a source vertex and iteratively selects the vertex with the smallest tentative distance. Updates the distances to neighboring nodes until all shortest paths from the source are found. *Difference:* Prim's focuses on minimizing total tree weight (spanning tree), while Dijkstra's minimizes path distance from a single source.

3. **Max-flow / Min-cut theorem:** The maximum possible flow from a source node s to a sink node t equals the total capacity of the smallest cut that separates s and t. Increasing flow saturates edges until no more capacity remains across some bottleneck set of edges and that bottleneck defines the minimum cut.

4. **Ford–Fulkerson method:** Start with zero flow and while there exists an augmenting path from s to t in the residual graph:

   (a) Find the path and determine its minimum residual capacity.

   (b) Increase the flow along that path by this capacity.

   (c) Update the residual network.

   Then repeat until no augmenting paths remain the final flow is the maximum flow.