Monte Carlo Algorithms for Quantum Systems

Mark Dubynskyi, Raghavendra Guggilam, Kyle Hess, Anthony Pizzimenti, Michael Jarret-Baume

Mason Experimental Geometry Lab

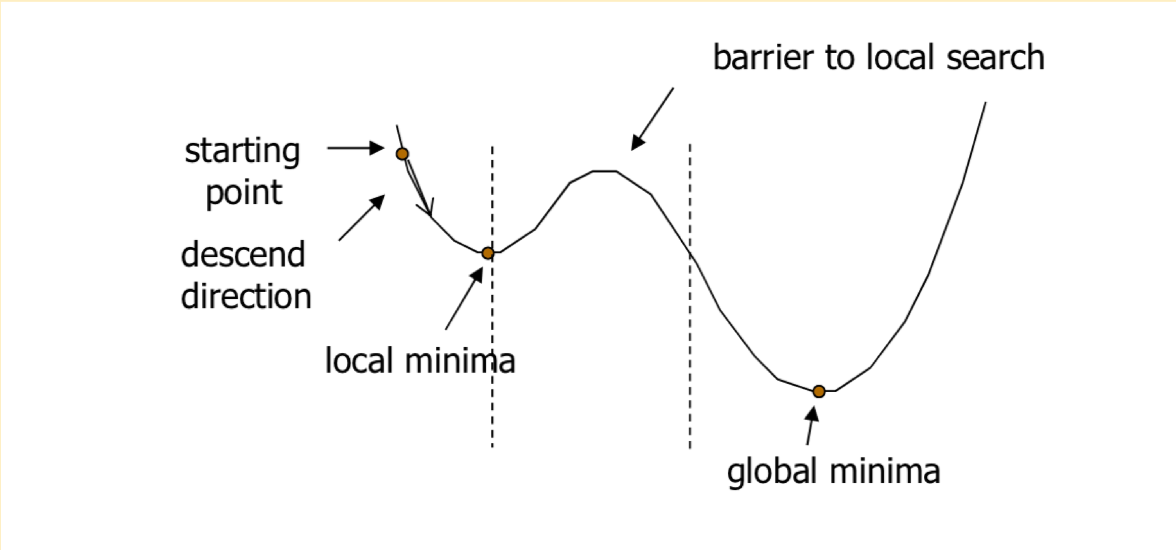
MEGL Final Symposium, December 2024

Quantum Monte Carlo Introduction

Our group dives deep into the notion of Quantum Monte Carlo for Quantum Algorithms. Quantum Monte Carlo (QMC) methods are a class of computational algorithms used to solve quantum problems where exact solutions are often not feasible due to their computational complexity.

Simulated Annealing

Simulated annealing is a metaheuristic for finding a good approximation of the global optimum of a given function. Simulated annealing mimics the physical process of heating and then slowly cooling a material, moving the system to neighboring states with an acceptance probability based on the energy of current state, the energy of a proposed neighboring state, and a temperature. The temperature cools periodically, reducing the likelihood of moving towards higher energy states, thus settling into a minimum-energy state.



Logic Synthesis/Majority Gate

One of the problems to which we apply parallel tempering is the Logic Synthesis problem [2]. This problem seeks to reduce the number of simple gates to recreate higher-level functions. In particular, we are trying to improve on the state of the art for synthesis of majority-n gates out of majority-3 gates for odd values of n. These circuits are represented using ternary graphs. The energy (or cost) of a state (one logic circuit) is determined by the number of inputs that result in an incorrect majority. We say that $E(N) = 0$ if, and only if,

$$f(x) = N(x) \quad \forall x \in \{0,1\}^n.$$

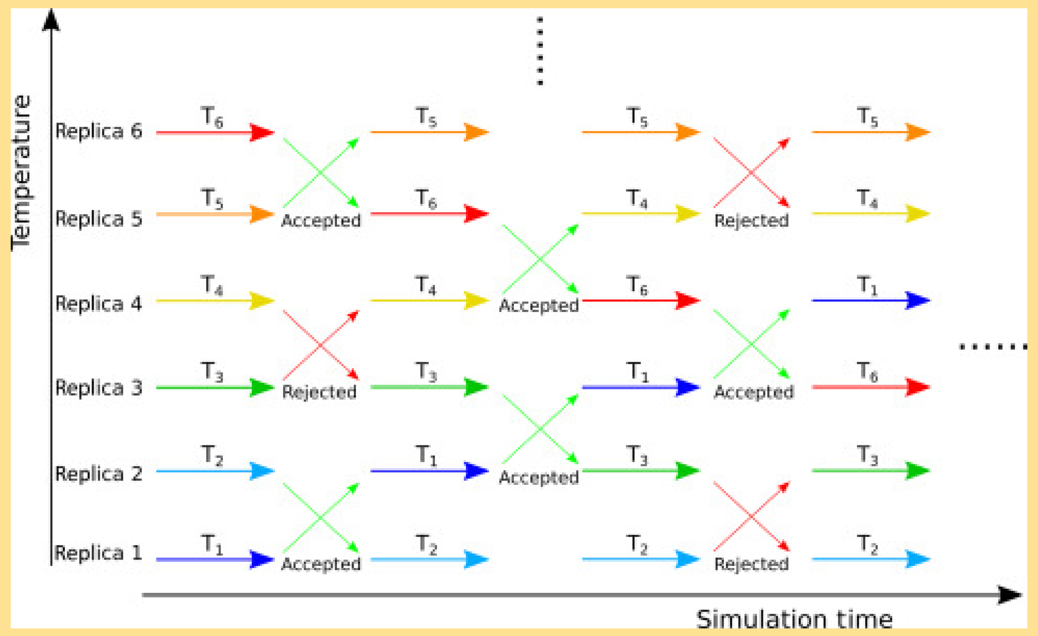
If $N(x)$ is the result of running an input through a network and f is the boolean value function to imitate, then the cost of a network is

$$E(N) = \sum_{x \in \{0,1\}^n} f(x) \oplus N(x).$$

Parallel Tempering

Overview

Parallel tempering is essentially an improved version of simulated annealing. Parallel tempering, (aka replica exchange Monte Carlo), is an advanced Monte Carlo simulation technique used to enhance the sampling efficiency of a system, especially when the system's energy landscape contains many local minima. This method involves running multiple simultaneous simulations (or "replicas") of the same system at different temperatures. Higher temperatures allow the system to overcome energy barriers and escape local minima, exploring the energy landscape more broadly.



Pseudocode

We formalized the parallel tempering algorithm by creating pseudocode based on existing work. Let T be a finite subset of the real numbers (representing the "energies" of the replicas), and let h be a stopping condition.

Algorithm 1 Parallel Tempering

```
function ParallelTempering(T, h)
    W ← empty array of length |T|
    w_i ← dist(V) for 0 ≤ i < |T|
    w_min ← w_i
    while !h(W) do
        for i ∈ {0, ..., |T| − 1} do
            u ← ProposeUpdate(w_i)
            w_i ← u with probability min{1, Cost(w_i, t)/
            if Cost(w_i) < Cost(w_min) then
                w_min ← w_i
        SORT(W)
    return w_min
```

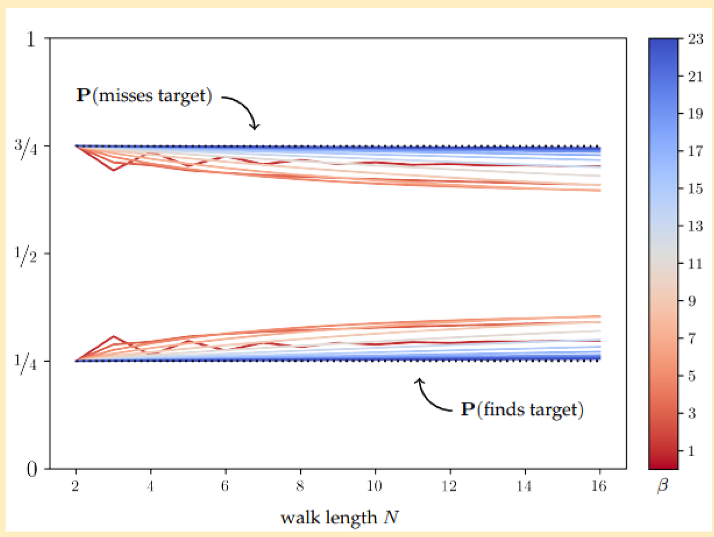
Advantage of PT over SA

Parallel Tempering (PT) is generally considered superior to Simulated Annealing (SA) because while SA relies on a single temperature schedule, PT exchanges information between multiple replicas at different temperatures, allowing high-temperature replicas to explore broadly and low-temperature replicas to refine solutions locally.
Key advantages of PT over SA include:

Exponential Efficiency: PT achieves an exponential advantage over SA in escaping local minima and reaching the global optimum, as it does not solely depend on a gradual cooling process.

Parallel Exploration: By running parallel chains at different temperatures, PT explores diverse regions of the search space, avoiding the limitations of a single trajectory as in SA.

Enhanced Sampling: PT effectively mixes between energy levels due to swaps between chains, ensuring better sampling of the solution space and improved convergence.

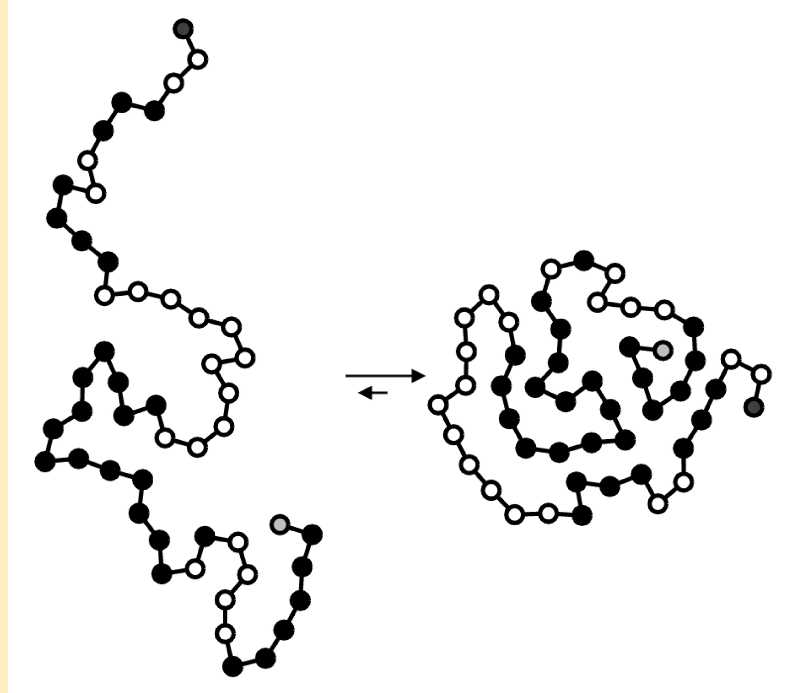


Test Cases/ Future Work

Code and Software Development: We have been writing code for Parallel Tempering to be used as a Python module. We are also working on program verification and optimization

Test Cases: We are developing new test cases such as the Ising Model simulation and the Protein Folder. Logic Synthesis.

Majority Gate: We hope to improve upon the state of the art for gates required for logic synthesis of majority-n gates.



Acknowledgments

We would like to give special thanks to Anton Lukyanenko, Swan Klein, and all those who run and support MEGL.

References

[1] P. Grassberger, "Go with the winners: a general Monte Carlo strategy," Computer Physics Communications, vol. 147, no. 1, pp. 64–70, Aug. 2002, doi: 10.1016/S0010-4655(02)00205-9.

[2] T. Häner, D. S. Steiger, and H. G. Katzgraber, "Parallel Tempering for Logic Synthesis." arXiv, Nov. 21, 2023. Accessed: Jan. 26, 2024. [Online]. Available: http://arxiv.org/abs/2311.12394

[3] M. Jarret and B. Lackey, "Substochastic Monte Carlo Algorithms." arXiv, Apr. 28, 2017. doi: 10.48550/arXiv.1704.09014.

[4] E. Testa, M. Soeken, L. G. Amaru, W. Haaswijk, and G. De Micheli, "Mapping Monotone Boolean Functions into Majority," IEEE Trans. Comput., vol. 68, no. 5, pp. 791–797, May 2019, doi: 10.1109/TC.2018.2881245.

[5] J.-S. Wang and R. H. Swendsen, "Replica Monte Carlo Simulation (Revisited)," Prog. Theor. Phys. Suppl., vol. 157, pp. 317–323, 2005, doi: 10.1143/PTPS.157.317.