Fast GPU Accelerated Ising Models for Practical Combinatorial Optimization

Omar El Nesr

Mentor: Axel Feldmann
October 15, 2023
Combinatorial Optimization

Finding the best solution from a finite set of possible solutions

Example:
Traveling Salesman Problem
Combinatorial Optimization

Finding the best solution from a finite set of possible solutions

Example:
Traveling Salesman Problem

1 - 4 - 3 - 2 - 1: 180
Combinatorial Optimization

Finding the best solution from a finite set of possible solutions

Example:
Traveling Salesman Problem

1 - 4 - 3 - 2 - 1: 180
1 - 3 - 2 - 4 - 1: 210
Combinatorial Optimization

Finding the best solution from a finite set of possible solutions

Example:
Traveling Salesman Problem

1 - 4 - 3 - 2 - 1: 180
1 - 3 - 2 - 4 - 1: 210
1 - 2 - 4 - 3 - 1: 170
Combinatorial Optimization

Finding the best solution from a finite set of possible solutions

Example:
Traveling Salesman Problem

1 - 4 - 3 - 2 - 1: 180
1 - 3 - 2 - 4 - 1: 210
1 - 2 - 4 - 3 - 1: 170

- NP-Hard $\rightarrow$ no known fast exact algorithms, but still want to solve
- Many applications, e.g. biotech & finance
- Solver needs to be flexible enough for many problems, but also structured enough to be efficient
Ising Model

A physical representation of interactions between magnetic particles

+1

-1
Ising Model

A physical and flexible representation of interactions between magnetic particles.

1 - 2 - 4 - 3 - 1: 170
MAXCUT

A problem that maps directly to the Ising Model

Cut: 5
MAXCUT

A problem that maps directly to the Ising Model

Cut: 6
Prior Work

Many attempts, but none are optimal

1. Simulated Annealing (SA): Flips spins one at a time until the cut is maximized. → Sequential, long runtime for poor solution quality

2. Parallel Tempering: Runs several (~8) SA instances in parallel. → Marginally superior to SA, but suffers from same problems

3. Simulated Bifurcation: Simulates a network of nonlinear optical oscillators → Higher solution quality, but current implementations are not optimized for real life problems
Simulated Bifurcation

Classical simulation of a quantum phenomena

- Initialize position (X) & momenta (Y) of oscillators
- Update X and Y for 0.1 seconds
- Calculate sum of interactions between all oscillators (matrix multiplication)
- Confine -1 ≤ Y ≤ 1
- Determine spin directions from X
- 10,000 iterations
Sparsity in Matrix Multiplication

Real life problems are sparse — we take advantage of this to get speedups

10x10 Dense Matrix: 10 multiplications per value

10x10 Sparse Matrix: 2 multiplications per value
GPU Computing

Extremely parallel solving
GPU Computing

Extremely parallel solving

Sequential Program Design
(CPU: easy, but slow)

Parallel Program Design
(GPU: hard, but fast)
Four Development Versions

Optimization steps to final algorithm

Dense Baseline
No optimizations

Sparse
Matrix multiplication redesigned to take advantage of sparsity

Fused Kernel
Fuses 3 separate steps (update, confine, interactions) into one GPU kernel

Graph
Chains all kernel calls together using CUDA Graph
Results: MAXCUT Speedup (Real Life)

Relative time to 10,000 steps for a representative sample of graphs

Gmean speedup: 14.5x faster
Max speedup: 58.1x faster

Results: MAXCUT Simulated Annealing Speedup

Comparison of Time-To-Solution for a representative sample of graphs

Speedup over Cook et al.

Gmean speedup: 25.5x faster
Max speedup: 44.7x faster

Comparison of Time-To-Solution for a representative sample of graphs

Speedup over Goto et al.

Gmean speedup: 3.3x faster
Max speedup: 2,318.6x faster

This is the fastest implementation we could find.

---

Results: Traveling Salesman Problem

Comparison of average TSP distances on available graphs

Red: Simulated Annealing
Blue: Zhang et al. (Simulated Bifurcation)
Green: My implementation

Average distance of TSP Tour. Lower is better.

All times: < 1 second
No time provided by other studies

Zhang, T., & Han, J. (2022). Efficient Traveling Salesman Problem Solvers using the Ising Model with Simulated Bifurcation.
Conclusions

This is the fastest Ising solver

Our algorithm is:

- On average ~3x faster...
- And up to ~2,000x faster than the previous leading implementation
- Open-source and free
- Adaptable to any combinatorial optimization problem
- 1000s of “agents” can be run simultaneously

Leading implementation costs $200 per hour for use on Amazon Web Services

Access to our algorithm is free