Quantum Annealing Simulation with Path-Integral Quantum Monte Carlo

Hadayat Seddiqi

Oak Ridge National Laboratory

hadsed@gmail.com

February 19, 2015

Overview

What is quantum annealing?

First, what is adiabatic quantum computation?

- Assuming zero-temperature, exploit quantum fluctuations to find the ground states of a Hamiltonian
- Start in a ground state of a non-commuting Hamiltonian, H_i and slowly decrease its contribution to the total Hamiltonian and increase the contribution of H_f, our final Hamiltonian, until it's all that's left
- In math, $\mathcal{H}(t) = a(t)H_i + b(t)H_f$
- $t \in \{0, \delta t, 2\delta t, \ldots, T\}$, and $a(t)$ and $b(t)$ are functions that specify the "annealing schedule"
- Slowly means adiabatic, and theoretically adiabaticity is only guaranteed in the infinite-time limit
- But who has that kind of time? Instead, we minimize the (Landau-Zener) transitions from ground to excited states to some tolerance

Excitation Gaps

Diabatic transitions become more likely as the gap shrinks, and less likely as it widens. Our adjustments to the Hamiltonian should be a function of the gap if we wish to maintain adiabaticity (to some tolerance).

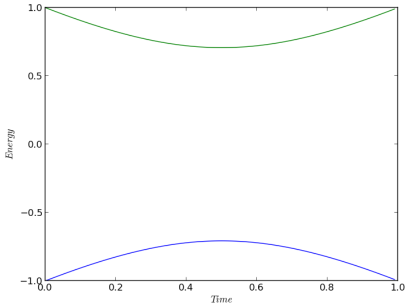

Figure: A one-qubit energy eigenspectrum. We should be slowing down as we approach $t = 0.5$.

Quantum Annealing Doesn't Care

Quantum annealing violates some AQC assumptions

- It is a heuristic, noisy, and more realistic implementation of AQC
- It assumes finite temperature, no guarantee of adiabaticity, and may not even necessarily care about ground states
- We will use the Ising spin-glass model in a transverse field as our time-dependent Hamiltonian

AQC Hamiltonian for Ising spin-glass in transverse field

$$\mathcal{H}(t) = -\sum_{i,j} J_{ij}\sigma_z^i \sigma_z^j - \sum_i h_i \sigma_z^i - \Gamma(t)\sum_i \sigma_x^i \tag{1}$$

Simulation Methods

- **Eigendecomposition of $\mathcal{H}(t)$ for some time-grid**
  - Requires storing $\mathcal{O}(2^{2k})$, i.e., the $\mathcal{H}(t)$ matrix for a particular t
  - Static simulator, no state vector and no dynamics
  - Advantage of that is that our steps can be larger depending on how much info we want, stability not an issue
- **Direct simulation of dynamics with the Schrodinger equation**
  - Requires storing $\mathcal{O}(2^{2k} + 2^k)$ for Hamiltonian and state vector
  - Dynamical simulation, means timestep must be chosen wisely for numerical stability
  - But we get probability coefficients for each eigenstate
  - And we can do zero-temperature, or add it artificially somehow
- **Path-integral quantum Monte Carlo**
  - Static estimator, so not exactly a dynamical simulation, but it does give rough population for each eigenstate
  - Only need to store $\mathcal{O}(k)$ floating-point numbers to track the state
  - Requires small but finite temperature
  - Can take exponential time to equilibrate for difficult problem instances

Path-integral quantum Monte Carlo

Implementing PIQMC

- Map a D-dimensional quantum system into a $D+1$-dimensional classical system and do Monte Carlo sampling on it
- Start system in a random state, copy over to each replica along this new "Trotter dimension"
- Couplings along Trotter dimension depend on magnetic field strength, weak in the beginning, strong in the end
- These two things are required to define a path integral, enforces all paths to start and end at same point but allows paths to vary in between

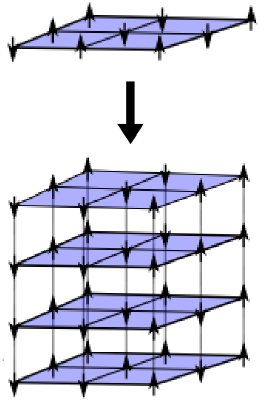
Figure: Troyer et al. (2014) arXiv:1411.5693

Path-integral quantum Monte Carlo

So the quantum Ising Hamiltonian

$$\mathcal{H}(t) = -\sum_{i,j}^{N} J_{ij}\sigma_z^i\sigma_z^j - \sum_{i}^{N} h_i\sigma_z^i - \Gamma(t)\sum_i \sigma_x^i \tag{2}$$

is transformed to a classical one with an extra dimension

$$\mathcal{H}_{d+1}(t) = -\sum_{k}^{P}\left(\sum_{i,j}^{N} J_{ij}z_i^k z_j^k + \sum_{i}^{N} h_i z_i^k + J_\perp \sum_{i}^{N} z_i^k z_i^{k+1}\right) \tag{3}$$

where

$$J_\perp = -\frac{PT}{2}\log\tanh\left(\frac{\Gamma}{PT}\right) > 0 \tag{4}$$

which goes off to infinity as Γ becomes small, and goes to zero as Γ gets large. Note the quantum spin σ is now a classical spin z.

Algorithm

Some pseudocode:

```
 1: $N \leftarrow$ number of spins in $\mathcal{H}$
 2: $P \leftarrow$ number of Trotter slices
 3: $T \leftarrow$ ambient temperature
 4: $\Gamma_{start}, \Gamma_{end} \leftarrow$ starting and final magnetic field strengths
 5: $\Gamma_{step} \leftarrow$ amount to decrease magnetic field in each timestep
 6: $\mathbf{z} \leftarrow \{z_0, z_1, \ldots, z_{N-1}\}$, random initial configuration
 7: $\mathbf{Z} \leftarrow \{\mathbf{z}_0, \ldots, \mathbf{z}_k, \ldots, \mathbf{z}_{P-1}\}$        $\triangleright$ store $P$ copies of $\mathbf{z}$
 8:
 9: for $\gamma$ in $(\Gamma_{start} : \Gamma_{end} : \Gamma_{step})$ do        $\triangleright$ simulate quantum annealing
10:     for $Z_i^k$ in $\mathbf{Z}$ do
11:         $\mathbf{Z} \leftarrow \textsc{Metropolis}(\mathbf{Z}, i + k, T)$
12:
13: $\mathbf{Z}_{solution} \leftarrow$ lowest energy replica amongst the $P$ Trotter slices
```

Metropolis Spin Updates

The Metropolis rule can be stated mathematically as

$$P(z, z') = \min \left\{ 1, \exp\left(\frac{E_z - E_{z'}}{\tau} \right) \right\} \tag{5}$$

for an attempted spin flip $z \rightarrow z'$. Temperature introduces noise into the system and allows it to escape suboptimal states by making the Boltzmann distribution more uniform. (For our purposes, it is held constant as we use quantum instead of thermal fluctuations to escape minima.)

```
1: function METROPOLIS($\mathbf{s}, j, \tau$)
2:     $\mathbf{s}' \leftarrow$ configuration after the spin $s_j$ is flipped
3:     $\Delta \leftarrow E(\mathbf{s}) - E(\mathbf{s}')$
4:     if $\Delta$ is positive or $e^{\Delta/\tau} > U_{random}$ then
5:         return $\mathbf{s}'$
6:     else
7:         return $\mathbf{s}$
```

Computational Bottlenecks

Where can we optimize for speed?

- Energy function evaluation
  - Evaluating entire system energy requires one matrix-vector multiply and one inner product, twice since we flip a spin and compare. For entire simulation this is $\mathcal{O}(4P(N + N^2))$ for each timestep.
  - Instead, calculate energy difference in the local field of each spin. This only requires multiplications equal to the number of neighbors, four in a 2D lattice, plus two for neighboring Trotter replicas
- Parallelize spin updates
  - Independently update each replica, communicating only between neighboring replicas
  - Partition along regular D dimensions and communicate across intra-replica boundaries only
  - Let spins be updated asynchronously at each timestep (taskpool style)
  - Assign processors an entire independent simulation for purposes of gathering statistics
- Binary encoding of states for x32 or x64 speedup by using bitwise operations

Code

Code for simulated and quantum annealing (Cython, OpenMP, MPI):

https://github.com/hadsed/pathintegral-qmc

Example (for simulated annealing)

```
import numpy as np
import piqmc.sa as sa

nspins = 64
rng = np.random.RandomState()
schedule = np.linspace(3.0, 1.0, 100)
sweeps = 10
spinvector = np.array([ 2*rng.randint(2) - 1
                        for k in xrange(nspins) ])
J = np.random.rand((nspins,nspins))
sa.Anneal(schedule, sweeps, spinvector, J, rng)
print(``Final state:'', spinvector)
```

References

R. Martonak, G. Santoro, E. Tosatti (2002)
Quantum annealing by the path-integral Monte Carlo method: The two-dimensional random Ising model
Phys. Rev. B, 66, 094203
doi:10.1103/PhysRevB.66.094203

T. Rnnow et al. (2014)
Defining and detecting quantum speedup
Science, 345, 420-424
doi:10.1126/science.1252319

T. Preis, P. Virnau, W. Paul, J. J. Schneider (2009)
GPU Accelerated Monte Carlo Simulation of the 2D and 3D Ising Model
J. Comp. Phys., 228, 4468-4477
doi:10.1016/j.jcp.2009.03.018

The End