IS 470 Final Report: Quantum Inspired Optimization Methods Research and Benchmarking

Lam Ying Sheng

Advisor: Prof. Lau Hoong Chin

November 2019

1 Introduction

1.1 Problem Statement

There are many intractable optimisation problems in important domains such as vehicle routing, urban logistics, air & sea port scheduling, pharmaceutical research, and medicine & hospitals which rely on solving hard optimisation problems. Finding global optima for these problems using general purpose solvers is a challenge because an exact solution cannot be proved to be found in less than exponential time.

The project aims to explore the performance of quantum inspired algorithms such as Simulated Quantum Annealing (SQA). By collating the results, a comparison will be done against Fujitsu's Digital Annealer for standard TSPLIB problems.

1.2 Objectives and Goals of Research

The objective of the project is to deliver benchmarked results of standard TSPLIB problems using Simulated Quantum Annealing against Fujitsu's Digital Annealer (DA). A problem instance using Western Sahara TSP problem from the World TSP Library will be used to tune hyperparmeters, so that we can better understand how the algorithm changes along with different parameters.

Through this research, we hope to better understand if quantum inspired algorithms can provide advantages as a general-purpose solver whether in terms of speed & scale, or in the quality of solution obtained. To this end, this research aims to contribute insight into the question: ”Can Quantum Annealing machines be simulated effectively on a classical computer?”

The work includes building a simulated quantum annealing package which aims to find the global optima of a given QUBO objective function by using Quantum fluctuations of the energy gap.

The code deliverable of this project include: a package implementing Simulated Quantum Annealing, and a QUBO to Ising problem converting function.

1.3 Research Motivation

In many optimisation problems, algorithms from the domain of physics are used as meta-heuristics to solve these problems ranging from Simulated Annealing, to Parallel Tempering. These are used because many systems in nature can be characterised by physical concepts such as free energy, and temperature. Through these techniques, the researcher can understand more fundamental properties about the system, and also traverse a larger set of solutions systematically.

To solve optimisation problems, we make use of algorithms that either do hill-climbing, or metropolis algorithms to search a solution space. In the context of a minimisation problem, such algorithms lie in the classical domain which seek the global minimum point by placing a solution at some point in the landscape, and allowing that solution to move based on local variations.

Classical Algorithms will avoid climbing hills that are too high, and are prone to leading the traveler into local minimums which are not the global minimum.

Quantum Annealing (QA) begins with the solution simultaneously occupying many coordinates, making use of the quantum effect of superposition. This idea makes use of the adiabatic theorem of computation and quantum tunneling to gradually increase the probability of finding the solution in states corresponding to deep minima, as annealing progresses.

Through this research, we hope to better understand if quantum inspired algorithms can provide advantages as a general-purpose solver whether in terms of speed & scale, or in the quality of solution obtained.

1.4 Timeline of work

Phase	Scope of Work	Duration
1	Understand workflow of the Digital Annealer Read through existing literature about formulating QUBO's and simulated quantum annealing / path integral monte carlo Understand about benchmarks to be used	Week 1 - 3 (3 weeks)
2	Start building and doing small tests of packages built. Build simulated quantum annealer package	Week 4 - 10 (6 weeks)
3	Compare and test the different optimizers while performing optimizations of the different classes of solvers	Week 11 - 13 (3 weeks)
4	Consolidation and presentation of results	Week 13 - 14 (2 weeks)

2 Introduction to Simulated Quantum Annealing

2.1 Why Simulated Quantum Annealing?

Whilst we don't have a quantum computer, we are able to simulate certain effects of a quantum system given certain conditions (e.g. system is represented by a stoquastic Hamiltonian, whereby off-diagonal entries are non-positive [2].

Simulated Quantum Annealing is a computational strategy used by many physicists in their attempt to answer whether true quantum annealing is indeed superior to one simulated on a classical computer. It is a subset of Quantum Monte Carlo methods, used to study systems described by Schrodinger's equation:

$$H(t)|\psi(t)>= i\hbar|\psi(t)>$$

In Simulated Quantum Annealing and Quantum Annealing, a cost function $f : \{0, 1\}^n \to R$ is associated with a Hamiltonian H_f such that the ground state of the Hamiltonian H_f corresponds to the tour that minimises f. To achieve H_f, we prepare a superposition of N Qubits in ground state in the X basis transverse field as such:

$$H_0 = -\sum_{i=1}^{N} \sigma_i^x$$

In SQA, we make use of this property of superposition to produce valid system states from the addition or superposition of several other states that are part of a system.

It can be done effectively in many ways, from:

- Variational Monte Carlo
- Path Integral Monte Carlo
- Diffusion Monte Carlo
- Green's Function Monte Carlo
- Hirsch-Fye Monte Carlo
- Gaussian Monte Carlo

In this research, we perform SQA through Path Integral Monte Carlo. This attempts to approximate quantum superposition by the parallel execution of different paths, by which we can effectively simulate the change of problem landscape [3].

Analogous to the annealing schedule given in Quantum Annealing [2] to lower the thermal state of our system to the ground state hamiltonian H_f:

$$H := H(s) = (1 - s)H_0 + sH_f$$

We make use of cooling to explore the different paths taken in a Path Integral Monte Carlo.

2.2 The Adiabatic Theorem

While the annealing occurs, a phenomena - quantum tunneling - allows the solution to be simultaneously 'correlated' or 'communicate' with other particles in the energy landscape. Quantum mechanically constructed and explored paths can explore regions unavailable to the classical path, making use of tunneling phenomena.

This enables the particle to pass through local valleys; moving to the lowest possible energy state in a given run. As a result, highly non-convex systems can be easily explored and solved to an acceptable margin of an optimal solution in reasonable amounts of time.

As we change s, the equilibrium thermal state (β) of the system will evolve according to the adiabatic theorem of computation [6] with the adiabatic parameter. This allows the system to remain in a state of internal equilibrium and maintain detailed balance.

Concurrently, the adiabatic theorem of evolution guarantees that if we lower s slowly enough ($\beta = \infty$), the system will remain in the ground state.

This is illustrated in Figure 1:

To make use of Quantum Annealing, we require quantum hardware which performs quantum tunneling by making use of a superposition of Qubit states. However, such hardware does not generalise well and is not available at reasonable cost. Hence, we attempt to simulate it on a classical computer using path integrals.

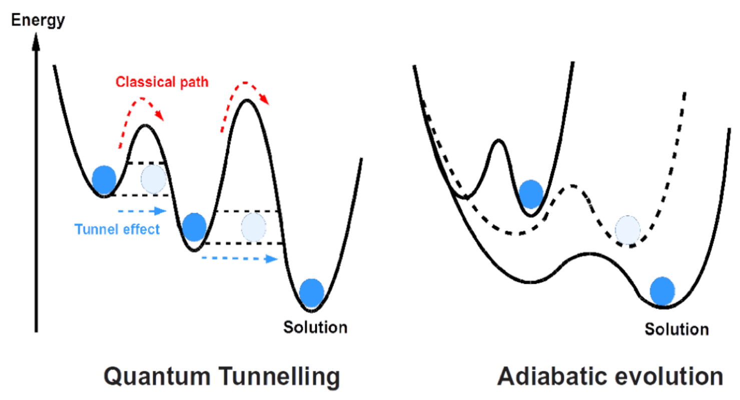


Figure 1: Illustration of the Adiabatic Evolution Process

2.3 The use of Path Integrals

Path integrals provide an intuitively appealing framework for expressing the thermal state of a system interpreted in quantum mechanics terms, and can be used as an expression for the probability amplitude of a system at the ground state. [7] We use Path Integrals to calculate the total energy of a solution across the solution space, whereby each path represents a route travelled.

We make use of Path Integrals because it can express the quantum partition function of a statistical mechanical system using a path integral. This provides a similar analogue to Simulated Annealing, whilst accounting for the relativistic weight of different paths along a tour. This accounts for the quantum nature of quantum annealing using a path integral factor.

To solve TSP problems by means of Path Integral Monte Carlo, this is done by expressing a path as a route, and performing monte carlo simulation of the system across a period of time. The path integral terminology is listed in appendix I.

Simulated Quantum Annealing makes use of Path Integral Monte Carlo by expressing a low temperature density operator (representing the state of the system,) as a product of high temperature density operators. We can approximate the thermal density matrix from one state to another state in a path integral expansion.

To approximate the final states of the system, we take the trace of the density states defined by σ, a density matrix:

$$\sigma(s) := e^{-\beta H(s)},\ Z(s) := tr(e^{-\beta H(s)})$$

The target function we try to approximate with Simulated Quantum Annealing, is the Quantum Partition Function $Z(s)$. It incorporates the quantum effects into the simulation using the concept of path integrals.

By mathematically splitting up the non-commuting terms using the Suzuki Trotter Expansion, we can approximate $Z(s)$.

We map the classical system into an Ising formulation. Since each city is now represented by n bits, each of the paths represent a route configuration for touring a city.

To approximate $Z(s)$, we express $Z(s)$ as a sum over an exponential number of nonnegative terms. SQA then proceeds by discretizing the adiabatic path and using the Markov chain Monte Carlo method to sample from π at various values of the adiabatic parameter $s_1 \approx 0; ... \ s_{max} \approx 1$.

For each trial move, a bit is randomly swapped with another bit and we calculate the difference in action resulting from the change in the bit from the previous time step to the current step.

The algorithm is covered in the next section.

2.4 Simulated Quantum Annealing Algorithm

1. Convert Objective Function (min Cost Function for TSP) into a Distance Matrix with city coordinates
2. Set the annealing parameters (generic): $s = 1.0$, $\beta = 10$, Geometric Parameter: 0.999, Trotters (L) = 10, MC Sweeps per Trotter = 30
3. Generate N paths randomly for each of L Trotters and initialize all spins to -1 except for one randomly selected city in each N, for each of L trotters
4. Perform MC moves, while trying to maintain the Boltzmann Distribution
5. Perform Acceptance / Rejection and use for the next update
6. Update s by geometric parameter
7. Go back to (3), until we reach $S \to 0$
8. Repeat Experiment N times
9. Marginalize results, get a probability distribution representing the likelihood of a solution

2.5 Quantum-ness of SQA

The quantumness of the algorithm is incorporated in trying to sample from the partition function across the entire algorithm. This sampling comes from having taking many samples of a particle exploring the energy landscape of the system we have specified.

At each point, a particle may have taken any combination of paths to reach it's final point, and to account for the relative weight of it's path against all possible paths taken by the particle we account for a flip using the $\tanh(\omega)$ factor.

It is accounted for below:

$$\frac{\pi(x')}{\pi(x)} = \sum_{x1,..,XL} \exp^{\frac{-\beta s}{L} \sum_{i=1}^{L}[f(x'_i - f(x_i)]} \frac{\prod_{j,k}^{n,L} tanh(\omega)^{|1_{x'_{j,k} \neq x'_{j,k+1}}|}}{\prod_{j,k}^{n,L} tanh(\omega)^{|1_{x_{j,k} \neq x_{j,k+1}}|}}$$

Hence, the final result at S = 0 will account for having explored a sufficiently large number of states in parallel through the world-line scheme.

From the algorithm, this is reflected in step (4) and step (5). A deeper explanation is given.

*4. Perform MC moves, while trying to maintain the classical Boltzmann Distribution**

1. For each Trotter path, randomly swap the path while maintaining the Hamiltonian cycle. Stitch together the route and calculate the energy of the route.
2. For each Trotter path, randomly swap the path while maintaining the Hamiltonian cycle
3. For each Trotter path, stitch together a route from each city visited

Whilst we are performing many simultaneous sweeps across the nL paths, we are actually trying to account for the relative weight of each path that the traveler could have taken

*5. Perform Acceptance / rejection and use for the next update**

1. Calculate the difference in action from each route. If the energy of the path is lower, accept the path and use it as for the next step. If the energy of the route is higher, accept the route with probability $\frac{1}{2nL}min(1, \frac{\pi(x')}{\pi(x)})$
2. If the route is rejected, resample from the routes and pick the one with the lowest route energy to partition function ratio.

When we perform acceptance or rejection of a certain path, we account for the difference in path from the previous step to the current step using the tanh term

2.6 Trotter Suzuki Decomposition

In a many particle system, the Hamiltonian H can be written as sum of kinetic and potential energy operators:

$$H = P + K$$

When we try to split up the terms of a Hamiltonian, we encounter a problem: Potential and Kinetic enrgy operators do not commute ($i.e. AB \neq BA$). We make use of the Trotter Suzuki Expansion to split up the non commuting terms, and obtain an error bounded by the order of L (the number of paths) we want to decompose the Hamiltonian into. [1]

To bound the error, we have to ensure that $\frac{\beta|H|}{L}$ needs to be sufficiently small or in other words, $L >> \beta|H|$. [2]

The first order of L is chosen for this simulation so that the partition function is $Z(s) = tr\ e^{A+B}$:

This gives us the mathematical expansion where:

$$Z(s) := e^{\frac{A}{L}} * e^{\frac{B}{L}} + \mathcal{O}(t)$$

where $A = \beta_s Hf$ and $B = -\beta(1-s)H_0$.

We can now make use of this result to compute the energy operators during the PIMC simulation.

3 Discussion of Results

3.1 Performance on the 29 City Western Sahara TSP

3.1.1 Own implementation

The simulation was first run over 100 iterations for Western Sahara [5] with 29 cities. The result is compared against the performance of PIMC versus a greedy algorithm, and the optimal solution.

From the results, we notice a few things:

1. SQA initially begins at an extremely high cost but very quickly improves and shows decreasing cost
2. At some point 0.8, the algorithm starts to perform better than the greedy algorithm, occasionally spiking above the cost of greedy
3. The algorithm then proceeds toward optimality, and it's best cost achieves a final margin of 5% from the optimal greedy solution

As the algorithm was run for a relatively short amount of time, it is likely the algorithm was still exploring different different configurations and trying to achieve optimality.

The results are shown in Appendix 4.2.

3.1.2 Using online algorithm as a comparison

A comparison was done with an online solver by WildQat [8] and it was found that the online solver performed poorly compared to the results by the implementation solver. It is likely the hyperparameters were not tuned.

However, this option was explored as I attempted to understand more about how a Hamiltonian based implementation would work and whether it would prove more effective than simulating PIMC dynamics through local swaps.

The results are presented in Appendix 4.4.

In comparison to DA, it was found that this algorithm performed slightly better than DA with a margin of 0.67% compared to DA's 2.27%. However, it was without tuning any hyperparameters.

3.2 Performance on TSPLib instances

The simulation was also run on over 20 TSPLib instances. They were run to demonstrate the performance on varying problem size versus the quality of solution obtained by PIMC.

The number of cities in the problem instance is beside the name of a city. So for instance, bier127 represents the bier problem instance for 127 cities.

In most cases, it was found that PIMC did not run better than the Greedy algorithm, and could not reach optimality. However in certain instances, especially for a smaller problem size (below 50 cities) PIMC performs within 10% margin of the optimal solution. The results are presented in the table in Appendix 4.3:

While PIMC may show poor performance for larger problem instances, it might be a useful algorithm to use for smaller problem sizes where cluster have already been formed.

In comparison to DA, it was found that DA performed better in all problem instances which results were presented for. This algorithm cannot be said to perform than DA for TSPLib problems.

It also can be seen from Appendix 4.6 that PIMC has a exponentially increasing run time compared to the DA. Although this may be attributed to improvements to the DA algorithm from clustering and running in parallel, it is undeniable the algorithm runs in exponential time. This poses an issue with running the algorithm on larger problem instances.

3.3 Hyperparameter Tuning

Several experiments with varying hyperparmeters were then carried out on the Sahara 29 City TSP. These were done by varying the following parameters (along with a brief explanation of the modification on the algorithm result:

- Beta (β) is the equilibrium thermal state of the system. It affects whether the system will remain close to the ground state. If $\beta = \infty$, the system will be exactly in the ground state while if β is small, the system will be far from the equilibrium state.
- L represents the number of Trotter paths from the Suzuki-Trotter decomposition. A higher L reduces the quality of final solution. This is because worse moves are accepted with lower probability. There exists a trade-off here.
- s Geometric Cooling Rate: The rate of cooling balances the inefficiency of the computation for a run of the algorithm while increasing the amount of states in the energy landscape explored. A slower cooling rate is likely to have an optimal solution found at the end of the algorithm.
- m Number of steps. Increasing the number of steps means the algorithm can explore a energy larger landscape for every path.

While there seems to be an optimal combination of $s, L, \beta, andNumberofsteps$, it seems more work is required to find the optimal hyperparameters for different problems due to varying degree of inoptimality of solutions found for different problem sizes as seen in testing the algorithm against different city sizes of TSPLib.

3.4 Limitations of PIMC

While we have explored several problem instances with the PIMC implementation of SQA, a few limitations have to be noted:

- PIMC can take exponential time to equilibrate for difficult problem instances. This is seen for the exponentially growing time in the section on testing with TSPLib problems. Increasing difficulty of problem will lead to a longer time for the algorithm to find a good solution.
- The idea of PIMC makes use of classical simulation. In Quantum Principle, the Hamiltonian of the problem is stoquastic and the off-diagonal elements are negative. This negates a sign problem and ensures the solution is a positive real number. With complex problems, we cannot guarantee off-diagonal elements are negative and the problem may not be able to be solved using PIMC.
- PIMC is built on classical Markov-chain dynamics which is in principle unrelated to the Schrödinger quantum dynamics [4] of a real QA device. This means certain properties of Quantum Annealing dynamics cannot be simulated.

3.5 Future Works

3.6 General Comments on the Algorithm

Currently, the algorithm makes use of local bit flips to modify the route taken by any given path. This may not preserve feasibility in more complicated situations and is not truly quantum. Annealing on Quantum Hardware may be able to overcome this problem, where constraints can be embedded in ancilla bits.

The algorithm makes use of many computations across the nL space of bits and paths to calculate the path integral. This makes the algorithm highly unsuitable for scaling, as it runs in exponential time. An illustration of the complexities are as follows:

- Space Complexity $\mathcal{O}(n^2 * L)$
- Time Complexity $\mathcal{O}(n * L * Steps) + O(2^{N)} \rightarrow O(2^N)$

These problems will potentially hinder the scalability of this algorithm. However, there is potential to incorporate some ideas from SQA such as system state perturbation into other classical algorithms in the future. This is an open research question and can be furthered in the future.

3.6.1 Improvements to Algorithm

- Currently not easily generalizable to other types of TSP (e.g. TSP with time Windows) due to algorithm not taking in QUBO. Subject of future research in incorporating constraints by cooling the Hamiltonian Directly
- Optimise for speed:
  - Binary encoding of states for x32 or x64 speedup by using bitwise operations
  - Parallelise Spin Updates
  - Independently update each replica, communicating only between neighboring replicas
- Let spins be updated asynchronously at each timestep (taskpool style)

- 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 Ising Structure, plus two for neighboring Trotter replicas.

3.7 Acknowledgements

This project was done in part using the Fujitsu Digital Annealer, with access provided by SMU.

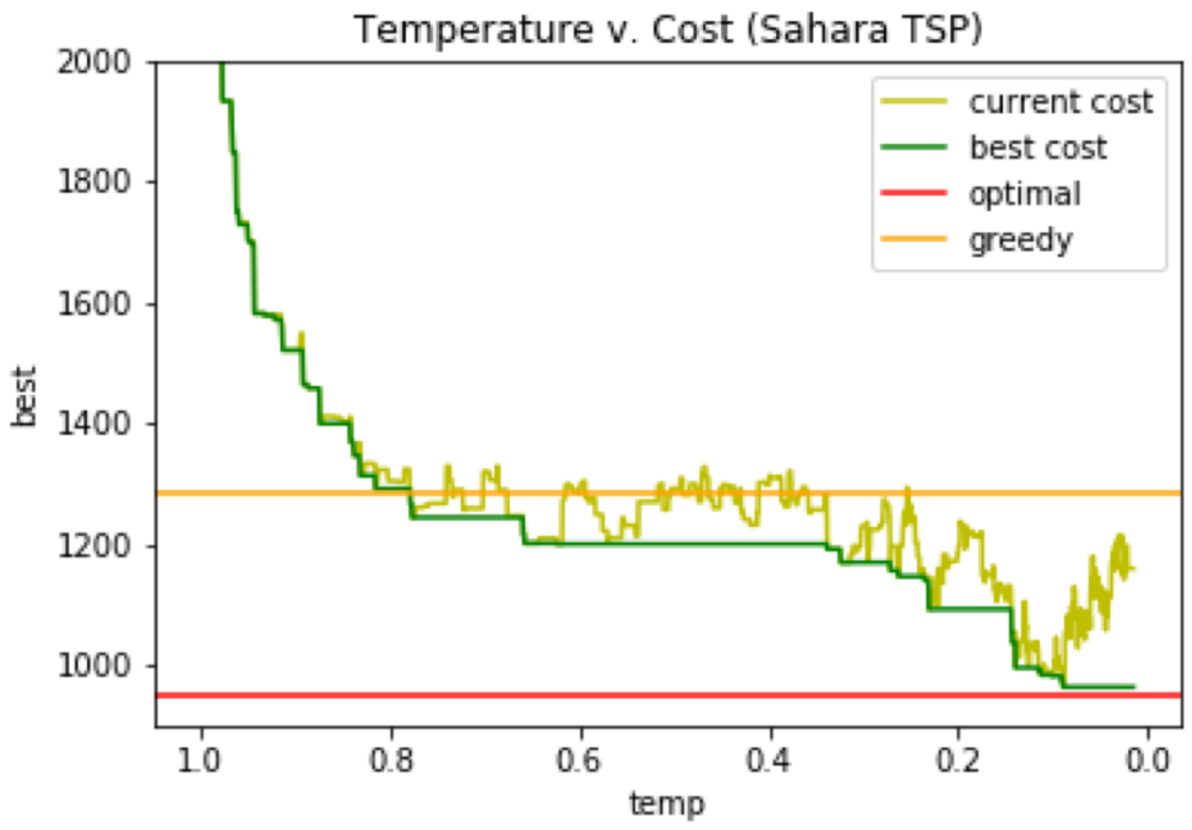
I would like to thank Prof. Lau Hoong Chin and the Digital Annealer team (Goh Siong Thye, Suen Whei Yeap, Sabrish Gopalakrishnan) under the SMU-Fujitsu Urban Computing and Engineering Centre for the support and insight they have brought to the table in supporting this exploration. There has been tremendous support for this project, and it would not have been possible without their assistance.

4 Appendix

4.1 Path Integral Terminology

Symbol	Explanation
L	Trotter Number (Number of paths explored)
N	Number of Bits used (# of paths)
Beta	Temperature of the System
S	Annealing Parameter

4.2 Results of PIMC Simulation

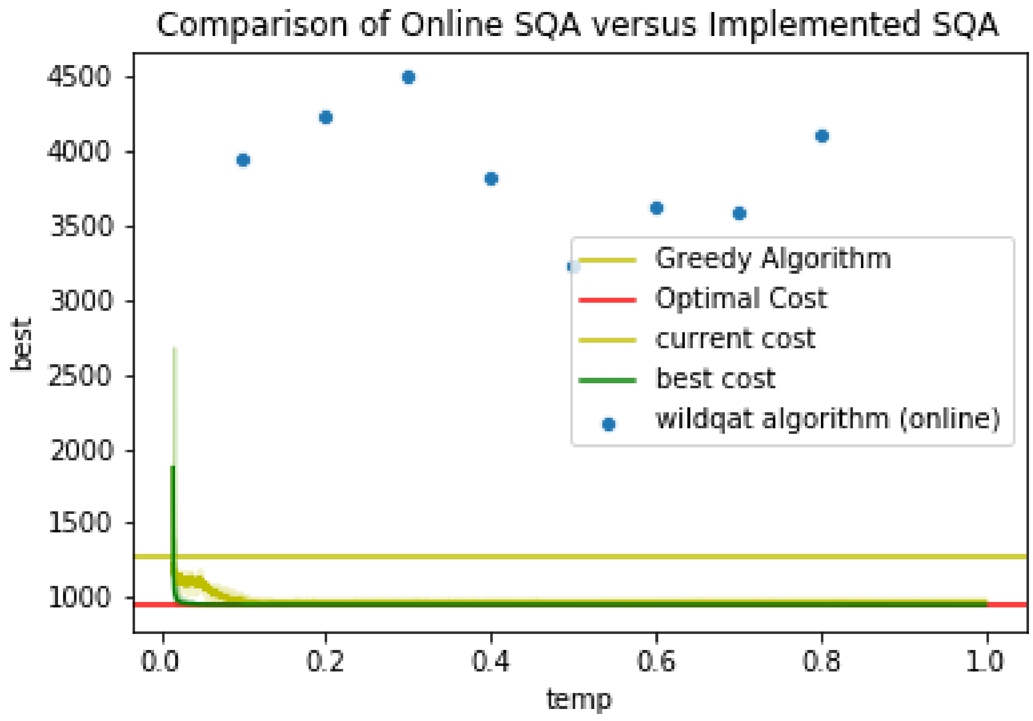


Steps	SQA Best	Margin (%)	Time	Greedy	DA Stitch	Margin (%)	Time	Best
100	952.47	0.67	661.5	1282.57	951.83	2.27	44.89	951.83

4.3 TSPLib Results

	problem_instance	best	da	da_margin	da_runtime	greedy	optimal	time
1	bier127	1766.00	1000.25	7.40	100.00	1104.99	931.35	5320.34
2	kroA100	391.15	223.89	5.20	70.00	252.55	212.82	3364.67
3	pr124	1165.79	522.31	9.70	116.00	561.36	476.05	5048.04
4	ulysses16	3.91	-	-	-	5.69	-	172.32
5	kroE100	385.26	228.04	3.30	72.00	268.83	220.68	5497.96
6	eil101	9.81	6.46	3.70	88.00	7.52	6.23	4706.16
7	kroD100	390.39	229.84	8.00	71.00	275.51	212.94	6232.30
8	rd100	136.70	-	-	-	103.74	-	4815.70
9	rat99	21.33	13.13	7.30	90.00	15.43	12.23	4411.73
10	gr137	12.31	-	-	-	7.54	-	5868.90
11	eil51	9.58	8.65	3.50	57.00	10.92	8.35	981.64
12	pr136	1460.11	774.80	8.90	124.00	874.13	711.56	5824.47
13	pr107	1012.53	474.92	14.70	98.00	476.29	414.05	3906.43
14	pr144	1161.69	546.00	34.00	104.00	425.64	406.51	6447.68
15	ulysses22	3.18	-	-	-	4.20	-	258.89
16	ch130	94.67	50.31	7.10	95.00	55.20	47.00	5524.26
17	eil76	9.68	7.30	3.20	83.00	9.04	7.08	2033.06
18	pr76	2128.33	1542.58	8.40	83.00	1872.77	1423.14	2023.71
19	gr96	8.55	-	-	-	7.28	-	3107.33
20	att48	797.99	-	-	-	853.58	-	885.33
21	lin105	266.23	150.35	9.80	84.00	199.00	136.94	5796.69
22	st70	13.37	10.04	4.20	68.00	11.43	9.64	1724.81
23	kroC100	389.49	222.18	7.10	71.00	269.19	207.49	5487.02
24	burma14	1.97	-	-	-	2.04	-	148.27
25	kroB100	401.52	231.08	4.30	73.00	279.43	221.41	5728.77
26	berlin52	181.11	-	-	-	161.21	-	1020.90

4.4 Online Solver



4.5 Hyperparameter Tuning Results

4.5.1 Changing Beta

beta	best	time	greedy	lowest	margin_sqa_greedy	margin_optimal
1	2948.94	16498.06	1282.57	951.82	229.92	309.81
10	2247.57	16167.92	1282.57	951.82	175.23	236.13
20	2004.58	16352.00	1282.57	951.82	156.29	210.60
30	1937.53	16392.78	1282.57	951.82	151.06	203.55
40	1906.20	16158.10	1282.57	951.82	148.62	200.26
50	1895.05	16128.71	1282.57	951.82	147.75	199.09
Average across Beta	2156.65	16282.93	1282.57	951.82	168.15	226.57

4.5.2 Changing Number of Steps

steps	best	time	greedy	lowest	margin_sqa_greedy	margin_optimal (%)
100	2695.91372	661.50	1282.57	951.82	210.19	283.23
1000	2089.53216	4448.76	1282.57	951.82	162.91	219.52
10000	1684.50685	43738.53	1282.57	951.82	131.33	176.97
Average across steps	2156.65091	16282.93	1282.57	951.82	168.15	226.57

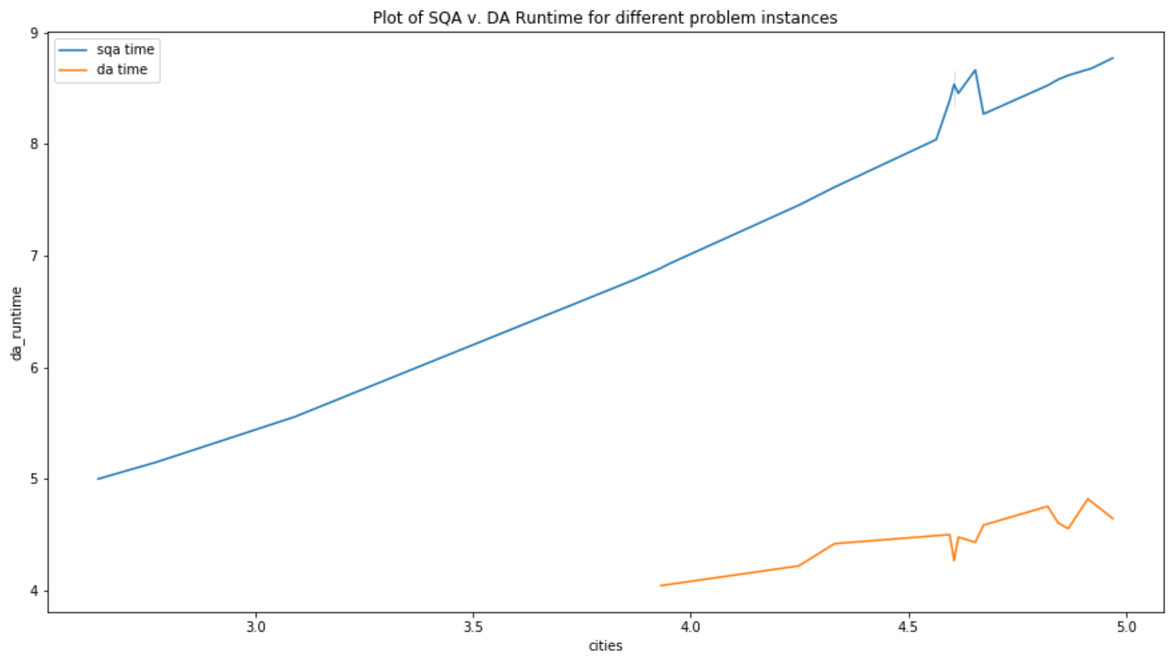
4.5.3 Changing Number of Trotter Paths

trotter	best	time	greedy	lowest	margin_sqa_greedy	margin_optimal (%)
5	2063.76	9227.16	1282.57	951.82	160.90	216.82
10	2103.88	14268.90	1282.57	951.82	164.03	221.03
20	2302.30	25352.72	1282.57	951.82	179.50	241.88
Average across Paths	2156.65	16282.93	1282.57	951.82	168.15	226.57

4.5.4 Changing the Cooling Rate

Geometric rate	best	time	greedy	lowest	margin_sqa_greedy	margin_optimal (%)
0.98	2053.83	1981.41	1282.57	951.82	160.13	215.77
0.99	1963.90	3999.48	1282.57	951.82	153.12	206.33
0.995	1831.56	7883.87	1282.57	951.82	142.80	192.42
0.999	1611.87	16240.90	1282.57	951.82	125.67	169.34
Average Geometric Rates	1644.48	15048.08	1282.57	951.82	128.29	172.87

4.6 Runtime of PIMC on TSPLib



References

[1] Jens Bonin Alexey Filinov and Michael Bonitz. Introduction to path integral monte carlo. part i. 2008.

[2] Aram W. Harrow Elizabeth Crosson. Simulated quantum annealing can be exponentially faster than classical simulated annealing. *Proc of FOCS 2016*, 2016.

[3] Luca Arceci Glen Bigan Mbeng, Lorenzo Privitera and Giuseppe E. Santoro. Dynamics of simulated quantum annealing in random ising chains. *PHYSICAL REVIEW*, B99, 2019.

[4] S. Isakov H. Neven S. Boixo Guglielmo Mazzola, M. Troyer. Simulated quantum annealing with quantum monte carlo. 2019.

[5] University of Waterloo. National travelling salesman problems.

[6] Martin Fraas Sven Bachmann, Wojciech De Roeck. Adiabatic theorem for quantum spin systems. *Phys. Rev. Lett. 119*, 060201, 2017.

[7] Vvedensky Westbroek, King and Durr. User's guide to monte carlo methods for evaluating path integrals. *American Journal of Physics*, 86, 2018.

[8] wildqat. Wildqat python sdk.