QUANTFOLD

Simulated Protein Folding with Quantum Annealing on D'Wave's Quantum Hardware for Drug Discovery Implications

Alice Liu

Project ID: **CBIO095**
Boston Latin School
Boston, MA, United States

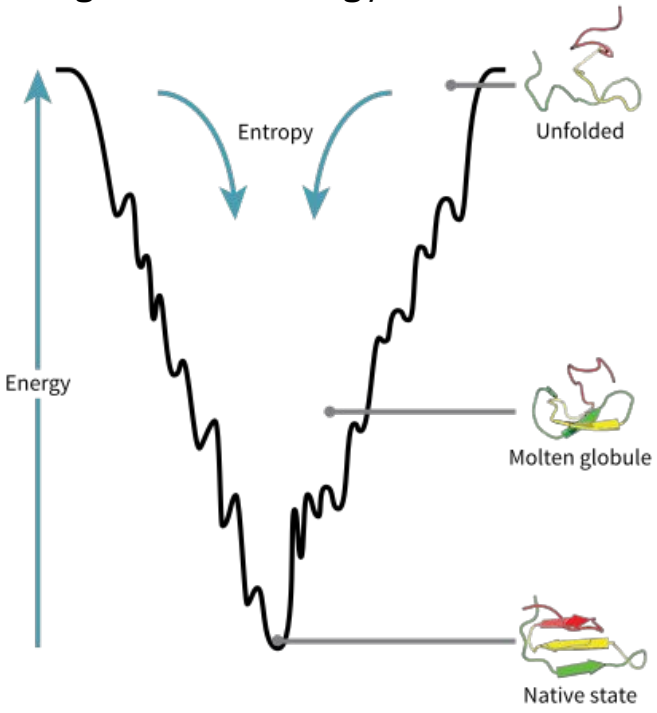
PROTEIN FOLDING

Proteins fold into their native structures by minimizing their free energy.

A protein's **native state** = its free energy minimum = properly folded or assembled form.

When proteins don't fold correctly to their lowest energy state, they are **misfolded** or denatured.

Misfolded proteins → negative effects.
Ex. **neurological diseases** such as Parkinson's, Alzheimer's and Huntington's stem from abnormally folded proteins



Energy Landscape

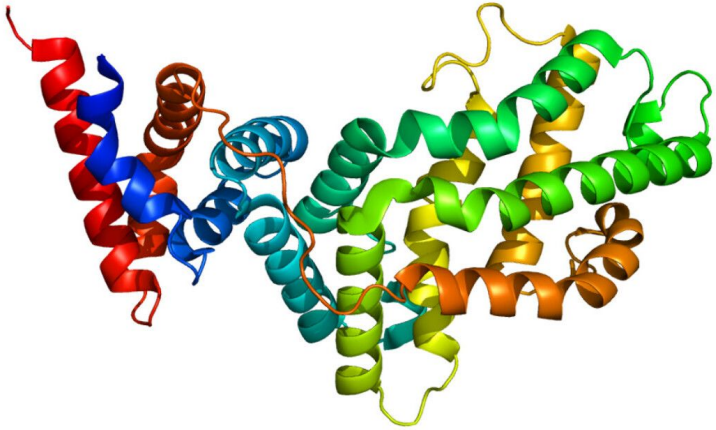
THE PROTEIN FOLDING PROBLEM

How can a protein's amino acid sequence dictate its three-dimensional atomic structure? [4]

Sequence

MPFGNTHNKFKLNYK
PEEEYPDLSKHNNHM
AKVLTLELYKKLRDKET
PSGFTVDDVIQTGVDN
PGHPFIMTVGCVAGDE
ESYEVFKELFDPIISDR
HGGYKTDSTEAF....

Structure



THE PROTEIN FOLDING PROBLEM

How can a protein's amino acid sequence dictate its three-dimensional atomic structure? [4]

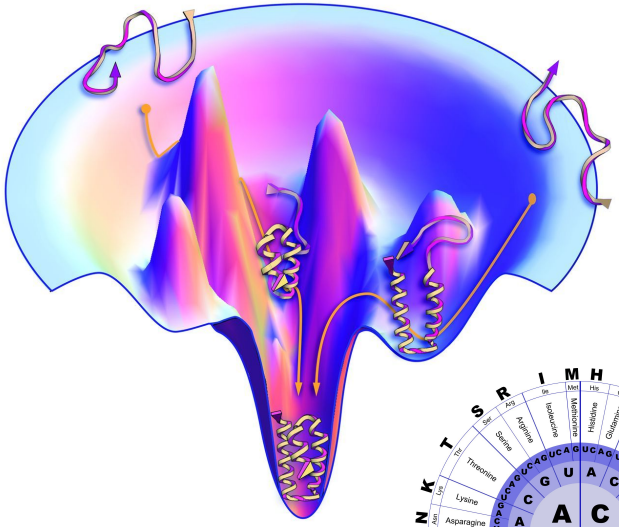
How can a protein fold so quickly despite a vast number of possible conformations (Levinthal's Paradox)? How does the protein know what conformations not to search?

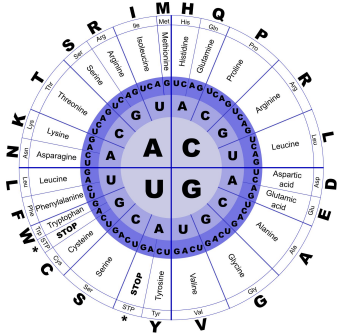
Is it possible to create an algorithm to predict a protein's native structure based on its amino acid sequence alone?

Knowing how proteins fold allows for the understanding in their 3D structure-function relationship in order to better understand enzymes and the treatment of misfolded-protein diseases[5].

COMPUTATIONAL (CURRENT) LIMITATIONS

Funnel Diagram of Energy Landscape





20 types of amino acids

Time and efficiency are both obstacles- our today's (classical) computers **lack the computational power** to effectively model the folding of these proteins.

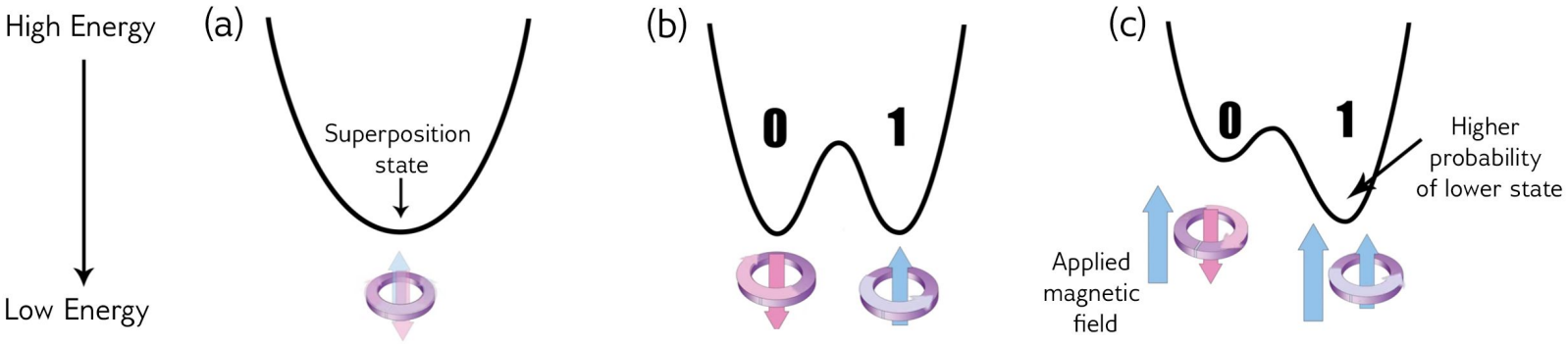
Proteins larger than 150 residues are unable to be computed classically.

No classical algorithm exists that is able to find the lowest energy state of a lattice protein within polynomial time.

QUANTUM ANNEALING

Qubits = quantum bit that can be in the state of 0, 1, or a **superposition** of 0 and 1 at the same time. (states 00, 01, 10, 11) , which allows for calculations to be run simultaneously.

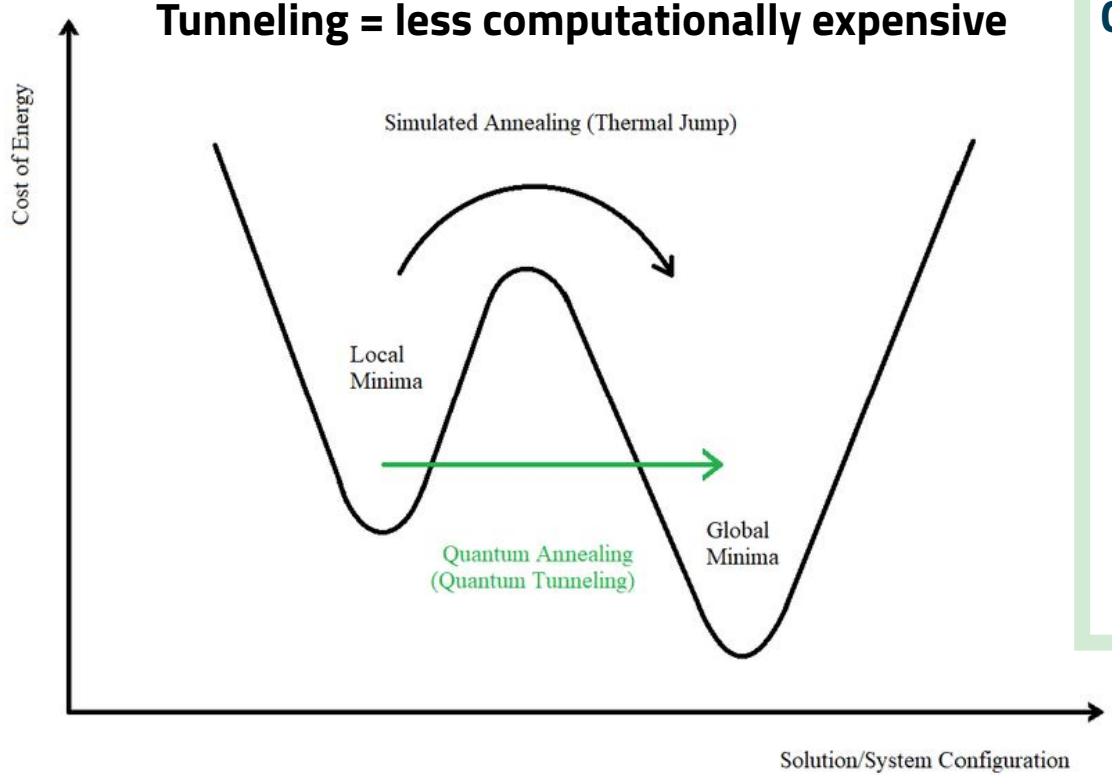
n qubits = 2^n solutions able to be run at the same time → speedup is ideal for optimization



- **Quantum Computing** helps choose a set of biases of couplers (affecting the final state of the qubits) that defines an energy landscape
- **Quantum Annealing** helps find the minimum energy of that energy landscape [7]

QUANTUM ADVANTAGE

Tunneling = less computationally expensive



Quantum Annealing

- Adaptive; works with a gradually decreasing parameter
- Not limited by barrier width and height
- On QPU: superposition of qubits with couplers and biases allow for efficient search where testing with the algorithm is completed simultaneously [6]

RESEARCH QUESTION

Can quantum computing methods speed up the process when finding the lowest energy conformations of an amino acid sequence?

HYPOTHESIS

The method that leverages quantum annealing with turn ancilla encoding on a regular CPU will yield the most effective results in terms of the highest number of instances that occured in the lowest energy conformations in the least amount of time.

This method is a hybrid of both classical simulated annealing and quantum annealing, where it can leverage the “speedups” within quantum annealing but will not be susceptible to noise and decoherence as experienced with quantum hardware when running on a QPU.

METHODS

EXPERIMENT OVERVIEW AND DETAILS

OBJECTIVE

Using quantum annealing on a quantum computer to explore its benefits in terms of speedups and increased accuracy (over classical computers) by simulating 2D lattice protein folding.

OVERVIEW

Tests the efficiency in the folding of amino acids in varying residue lengths in 3 parts:

1) Simulated Annealing with Conventional Monte Carlo
2) Quantum Annealing with Turn Ancilla Encoding on a CPU
3) Quantum Annealing on D'wave's quantum computer (QPU)

Within each of the three parts, 3 different types of amino acid sequences will be tested:

1) 6 Residues
2) 9 Residues
3) 12 Residues

What will be measured in each trial:

1) Number of lowest energy conformations achieved
2) Time to reach the minimum energy state

EXPERIMENT OVERVIEW AND DETAILS

VARIABLES: INDEPENDENT, DEPENDENT, CONTROL

Independent: Residue Length (either 6, 9, or 12 residues), Type of Method Tested (Quantum Annealing on CPU or Quantum Annealing on QPU)

Dependent: Number of instances in the lowest energy conformations for each amino acid, Time it takes to each the minimum energy state

Control Variable: Simulated Annealing with Conventional Monte Carlo (method that is currently being used today)

MATERIALS

- D'wave's Leap API Integration for 2000-Qubit Annealer
- Modern Operating System consisting of x86 64-bit CPU (Intel / AMD architecture) with at least 4 GB RAM and 5 GB free disk space. For this experiment, the Mac OS Mojave Version 10.14.5 with a 1.1 GHz Intel Core m3 was used.
- Python Version 3.7.4
- Interactive web-based computational environment, ex. Jupyter Notebook was used.

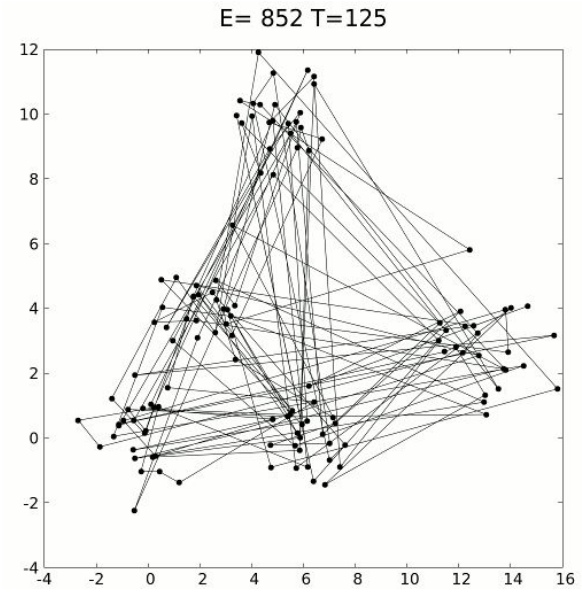
ALGORITHMS – SIMULATED ANNEALING [1]

Approximates the global optimum of a function with the slow decrease in the probability of accepting worse solutions as the solution space is explored

1. Define a schedule for annealing temperature **T**
2. Randomly choose its residue $\mathbf{r_i}$
3. Perform random walk, with respect to $\mathbf{r_{(i-1)}}$
4. Compute energy change in energy **ΔE** = E - E'
5. Accept step if **exp(-ΔE/T)** expresses the probability of a state of energy E relative to the probability of a state of zero energy > **random.uniform(0,1**).
   If ΔE <=0, always accept

$$\text{Probability} = \frac{1}{1+e^{\frac{-\Delta E}{T}}}$$

6. Process repeats as T ⟶ 0

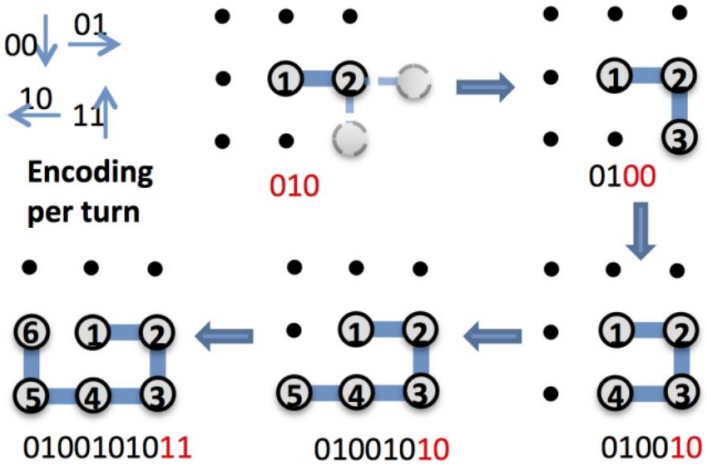


Simulated Annealing to Find Optimum

[1]Dimitris Bertsimas, John Tsitsiklis. "Simulated Annealing." Statist. Sci. 8 (1) 10 - 15, February, 1993. https://doi.org/10.1214/ss/1177011077

ALGORITHMS - TURN ANCILLA ENCODING [2]

Simulates **random walks** within a 2D lattice by adding constraints and requirements on where the protein folds. This is during the process of minimizing the objective function when finding the lowest energy conformations.



Binary representation of a 6-residue lattice protein in turn ancilla encoding.

Two qubits are needed per bond and the turn direction are denoted by 00 (downards), 01 (rightwards), 10 (leftwards) 11 (upwards)

[2] Perdomo-Ortiz, A., Dickson, N., Drew-Brook, M. *et al.* Finding low-energy conformations of lattice protein models by quantum annealing. *Sci Rep* **2,** 571 (2012). https://doi.org/10.1038/srep00571

ALGORITHMS - TURN ANCILLA ENCODING [8]

Simulates **random walks** within a 2D lattice by adding constraints and requirements on where the protein folds. This is during the process of minimizing the objective function when finding the lowest energy conformations.

$$E(q) = E_{back}(q) + E_{overlap}(q) + E_{pair}(q)$$

$E_{back}(q)$ → Penalizes protein fold if back-to-back when 2 consecutive edges go in between the same pair of vertices

$E_{overlap}(q)$ → Penalizes protein fold if lattice protein folds over itself

$E_{pair}(q)$ → Marks interaction between non-bonded acids adjacent on a lattice

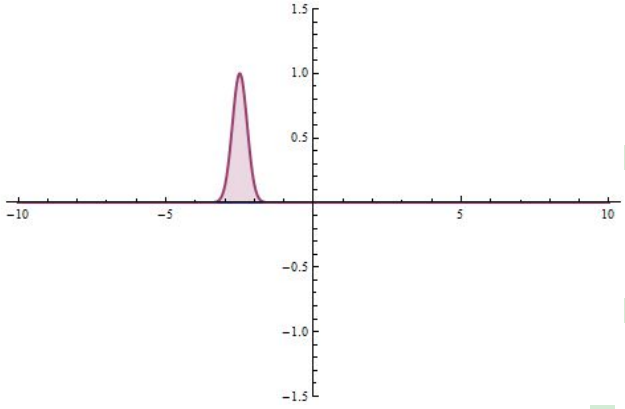
[8] Babbush, Ryan, et al. "Construction of Energy Functions for Lattice Heteropolymer Models: A Case Study in Constraint Satisfaction Programming and Adiabatic Quantum Optimization." *ArXiv.org*, 11 June 2013, arxiv.org/abs/1211.3422.

ALGORITHMS – QUANTUM ANNEALING[3]

Technique for finding global minimum of a given objective function over a set of candidate states by using quantum computation methods (quantum tunneling), which allows for speedups over its classical counterpart

1. Define a schedule for annealing temperature **T**
2. Randomly choose **ith qubit** q_i
3. Perform a qubit flip
4. Compute energy change in energy **ΔE** = E - E'
5. Accept step if **exp(-ΔE/T)** expresses the probability a state of energy E relative to the probability of a state of zero energy > **random.uniform(0,1**).
   If ΔE <=0, always accept

   Probability = $\frac{1}{1+e^{\frac{-\Delta E}{T}}}$
6. Process repeats as T ⟶ 0

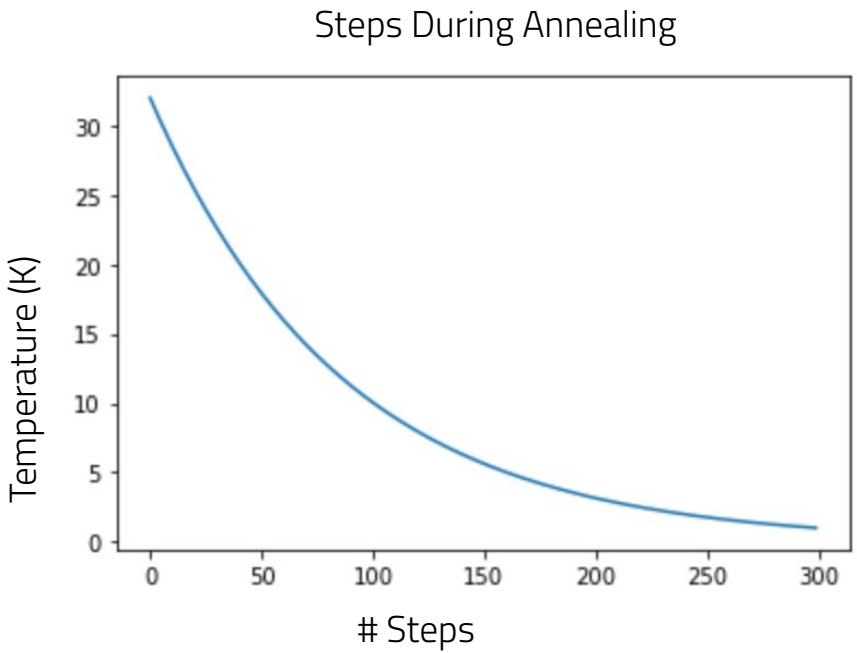


Quantum Tunneling effect during annealing allows the passage through an energy barrier (instead of going up and around the barrier with simulated annealing)

[3] Robert, A., Barkoutsos, P.K., Woerner, S. *et al.* Resource-efficient quantum algorithm for protein folding. *npj Quantum Inf* **7,** 38 (2021). https://doi.org/10.1038/s41534-021-00368-4

RESULTS

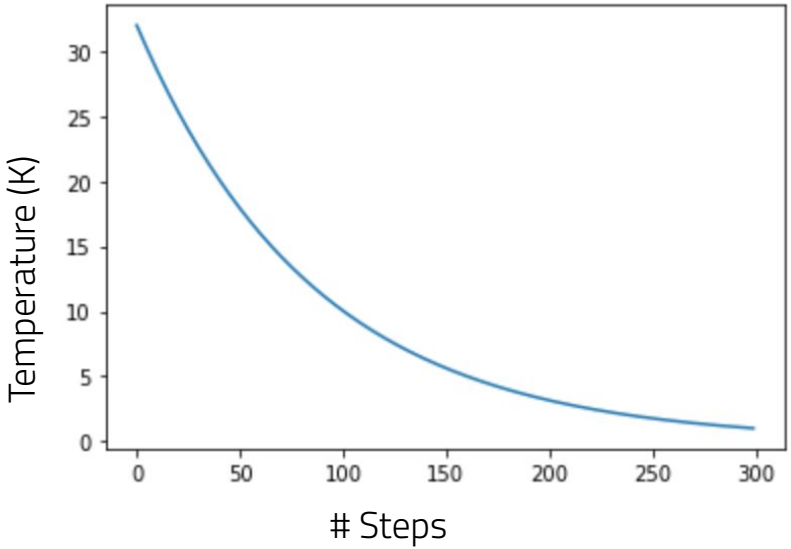
ANNEALING SCHEDULE + ENERGY LANDSCAPE



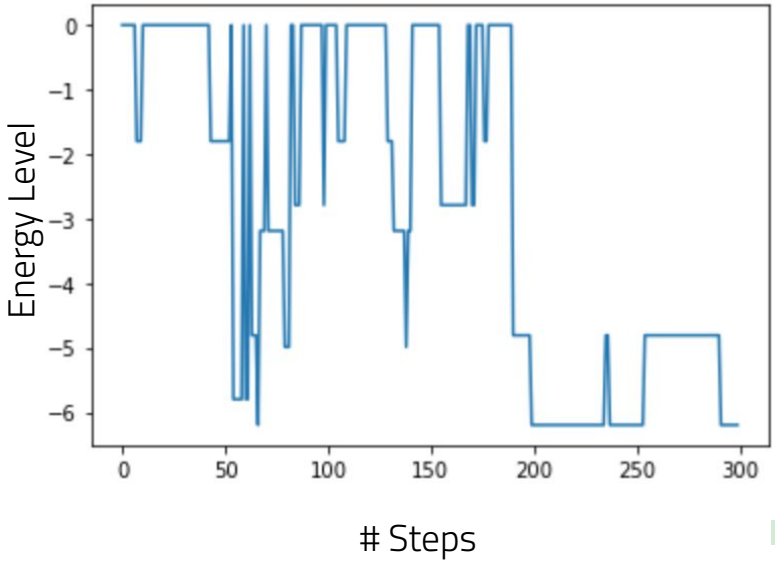
Quantum annealing finds the "worst solutions," ie. neighboring solutions that increase the state energy. Those solutions are eliminated in order to find the optimal.

ANNEALING SCHEDULE + ENERGY LANDSCAPE

Steps During Annealing



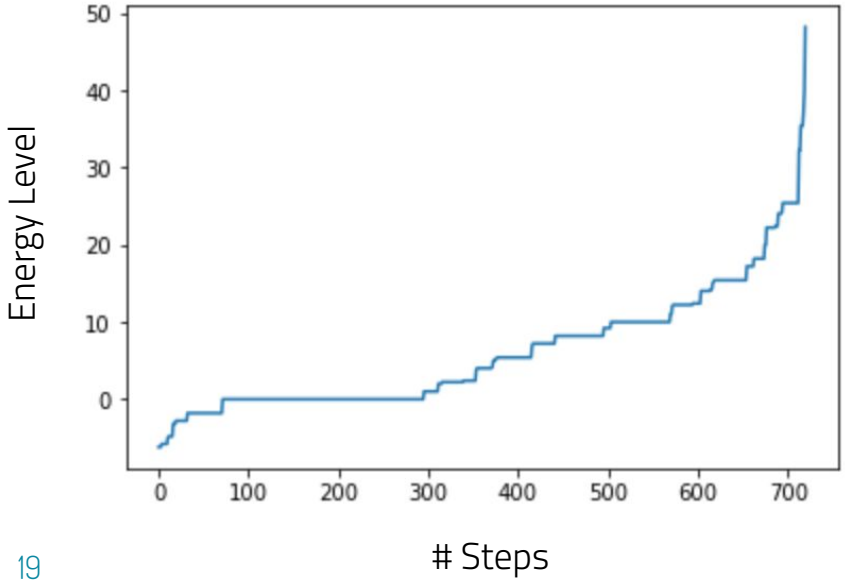
Energy Landscape - Simulated Annealing



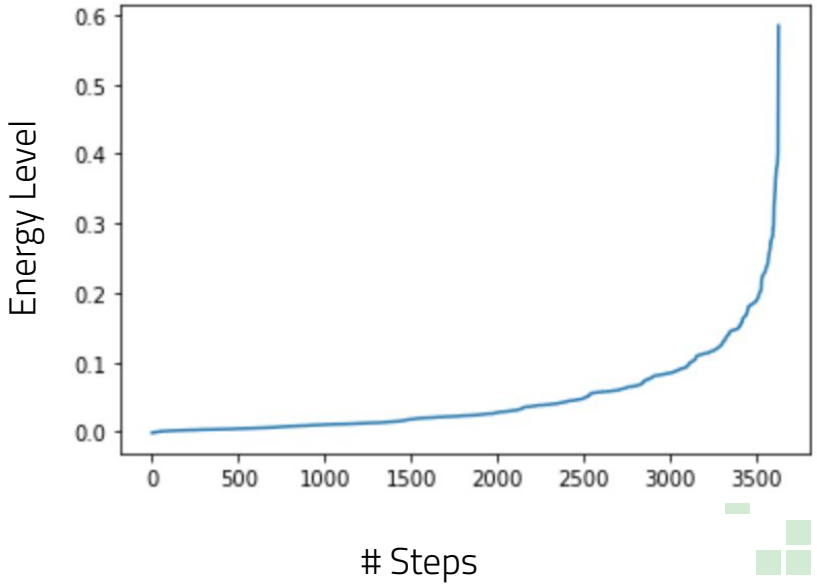
ANNEALING SCHEDULE + ENERGY LANDSCAPE

Quantum annealing finds the "worst solutions," ie. neighboring solutions that increase the state energy. Those solutions are eliminated in order to find the optimal.

Energy Landscape - Quantum Annealing with Turn Ancilla Encoding

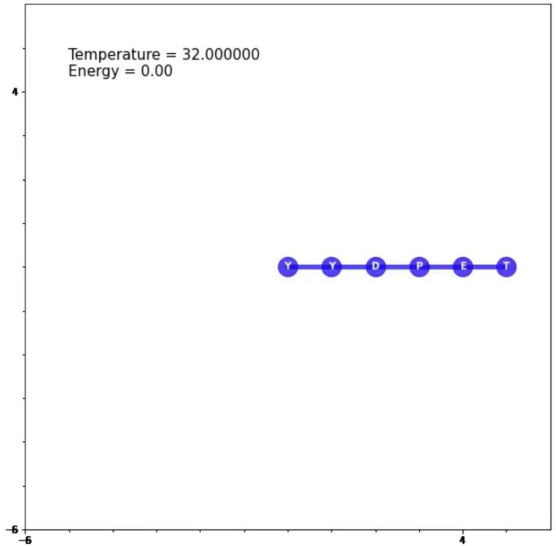


Energy Landscape - Quantum Annealing with D'wave's Annealer



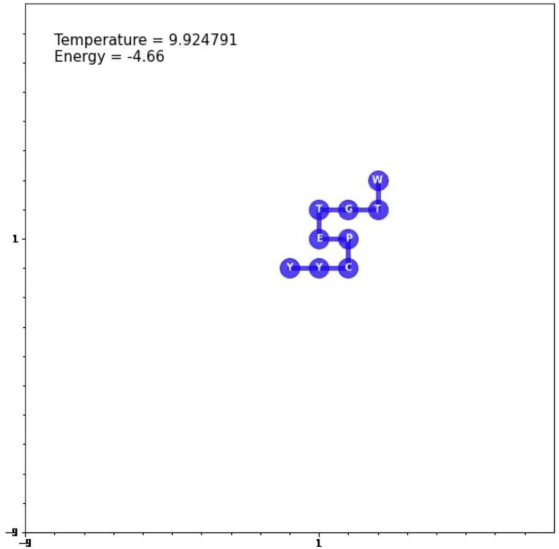
SIMULATED ANNEALING WITH MONTE CARLO (CONTROL)

6 Residues



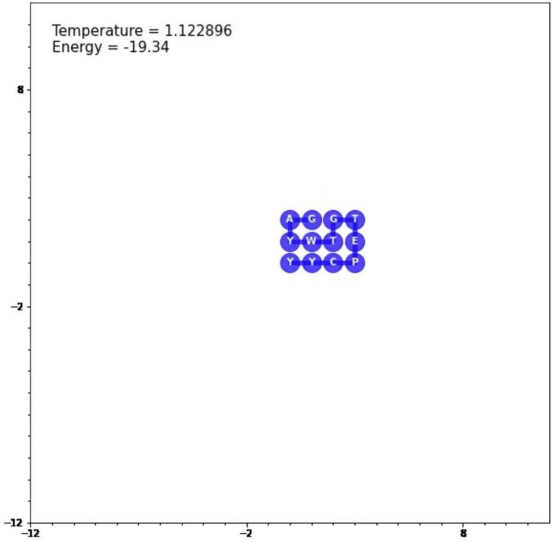
Lowest energy conformation:
67/360 = **18.6%**
Runtime: ~**12.2 seconds**

9 Residues



Lowest energy conformation:
48/360 = **13.3%**
Runtime: **~25 seconds**

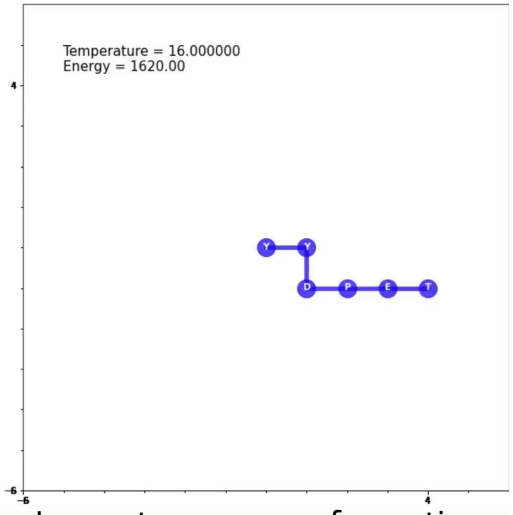
12 Residues



Lowest energy conformation:
22/360 = **6.1%**
Runtime: **~38 seconds**

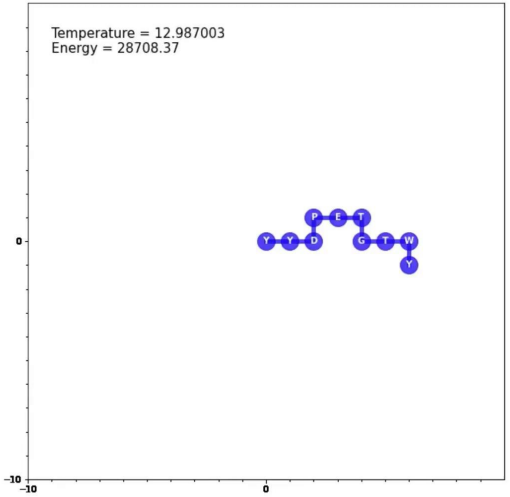
QUANTUM ANNEALING WITH TURN ANCILLA ENCODING

6 Residues



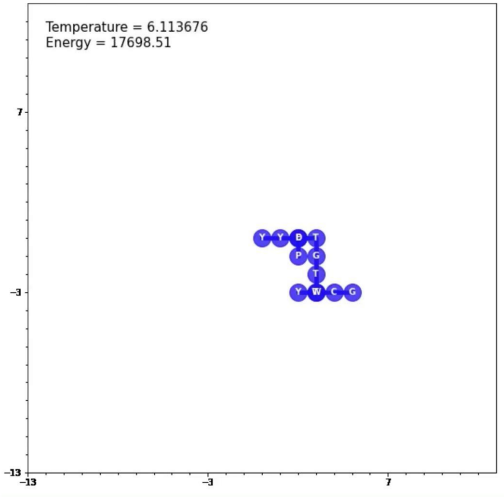
Lowest energy conformation:
625/720 = **86.6%**
Runtime: ~**8 seconds**

9 Residues



Lowest energy conformation:
510/720 = **70.8%**
Runtime: **~12 seconds**

12 Residues

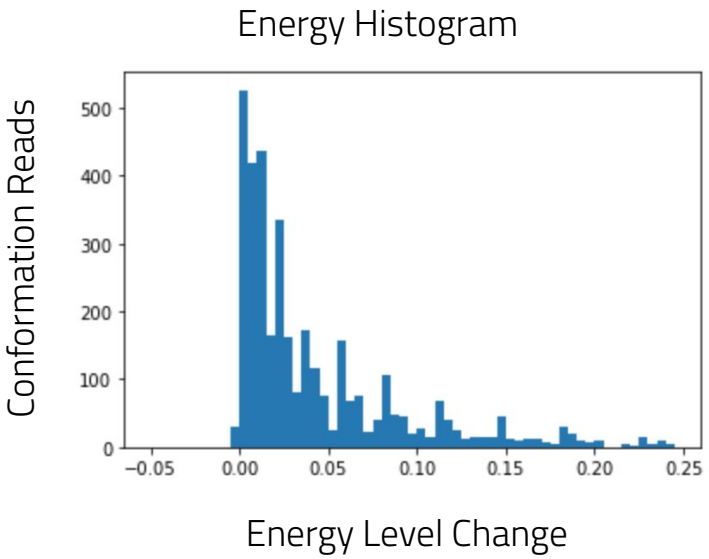


Lowest energy conformation:
440/720 = **61%**
Runtime: **~24 seconds**

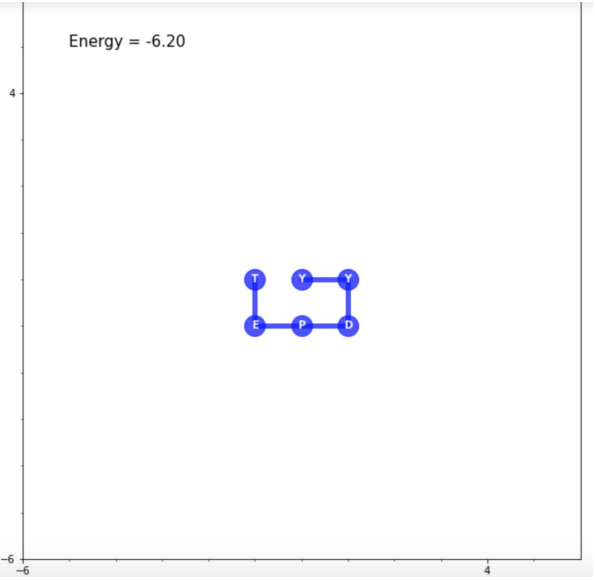
QUANTUM ANNEALING WITH DWAVE'S QPU

```
QPU time used: 262 ms or 0.262 ms/read
mean: 4.64, sd: 1.93
[-0.0014 -0.0014 -0.0014 -0.0013 -0.0013 -0.0013 -0.0013 -0.0013 -0.0013]
[-0.0014 -0.0014 -0.0014 -0.0013 -0.0013 -0.0013 -0.0013 -0.0013 -0.0013]
```

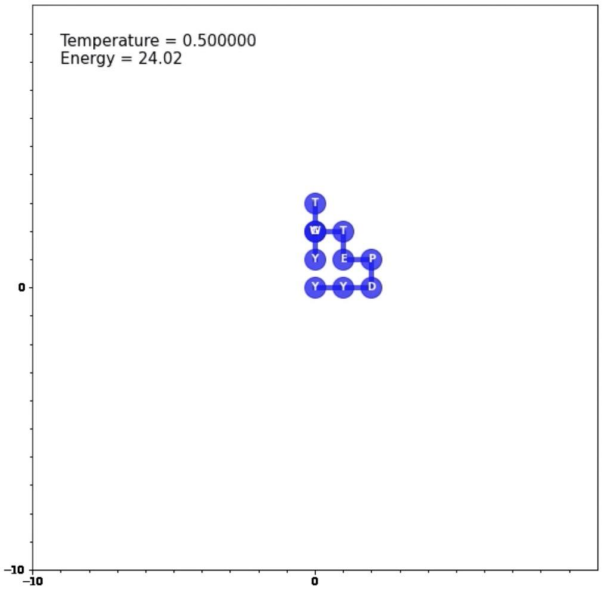
	state	isvalid	energy	Hb	Ho	Hi	count
0	1011011100111000110	True	-1.391382e-03	0	0	-6.20	3
1	1111011101111110000	True	-1.301616e-03	0	0	-5.80	8
2	0111011101111100000	True	-1.301616e-03	0	0	-5.80	1
3	1011011101100000010	True	-7.158887e-04	0	0	-3.19	1
4	1111011101111010000	True	-6.754937e-04	0	0	-3.01	3
5	0110011100110100000	True	-6.261221e-04	0	0	-2.79	1
6	0010001100110111100	True	-6.261221e-04	0	0	-2.79	2
7	1111011100110110100	True	-6.261221e-04	0	0	-2.79	2
8	1111011101110110000	True	-6.261221e-04	0	0	-2.79	4
9	0001000101110001101	True	-4.039497e-04	0	0	-1.80	1
10	0111011001110011001	True	-4.039497e-04	0	0	-1.80	1
11	1111011101111110001	True	-8.976661e-05	0	0	-0.40	1
12	0001000101110001100	True	-2.081668e-16	0	0	0.00	1
13	0000011000000000000	True	-1.387779e-17	0	0	0.00	1



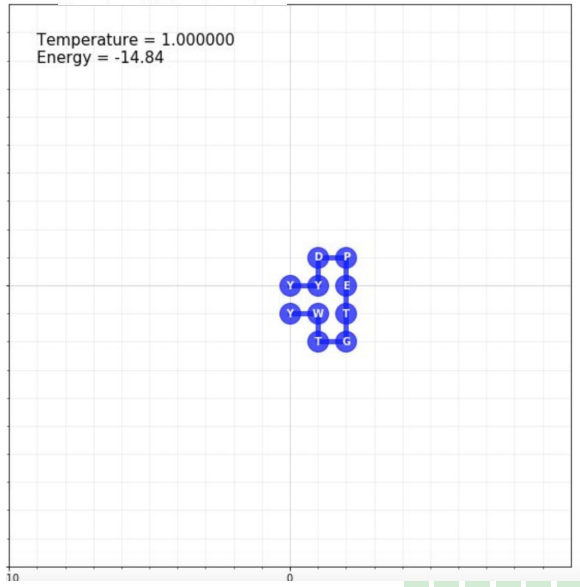
QUANTUM ANNEALING WITH DWAVE'S QPU



Lowest energy conformation:
538/720 = **74.7%**
Runtime: ~**2.62 seconds**



Lowest energy conformation:
449/720 = **62.3%**
Runtime: **~8 seconds**



Lowest energy conformation:
370/720 = **51.4%**
Runtime: **~12 seconds**

RESULTS OVERVIEW

Percentage of **instances** reaching lowest energy conformation

	6 Residues	9 Residues	12 Residues
Conventional Simulated Annealing	**18.6%**	**13.3%**	**6.1%**
Quantum Annealing, Turn Ancilla Encoding	**86.6%**	**70.8%**	**61.0%**
Quantum Annealing, D'wave's Annealer	**74.7%**	**62.3%**	**51.4%**

RESULTS OVERVIEW

Runtime (in seconds) to reach the minimum energy state

	6 Residues	9 Residues	12 Residues
Conventional Simulated Annealing	**12.2 s**	**25.0 s**	**38.0 s**
Quantum Annealing, Turn Ancilla Encoding	**8.0 s**	**12.0 s**	**24.0 s**
Quantum Annealing, D'wave's Annealer	**2.6 s**	**8.0 s**	**12.0 s**

DISCUSSION

RESULTS ANALYSIS

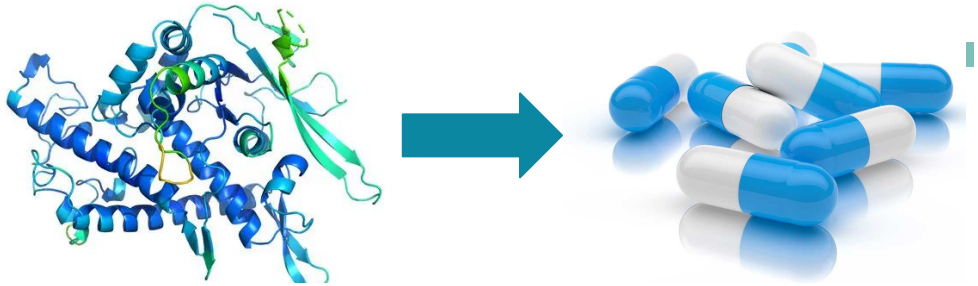
Highest number of instances in successful energy conformations: **Quantum Annealing with Turn Ancilla Encoding**
Lowest time to reach the minimum energy state: **Quantum Annealing with QPU**

1. The hypothesis is mostly supported by the experimental results. The quantum annealing method on a QPU with D'wave's Annealer yielded the lowest time to reach the minimum energy state across all residue lengths tested.

2. The quantum annealing method with turn ancilla encoding on a CPU yielded the highest percentage in the number of total successful energy conformations across all residue lengths tested.

3. The simulated annealing method (control) had the smallest yield, having the lowest percentage in the number of total successful energy conformations as well as the longest time to reach the minimum energy state across all residue lengths.

IMPLICATIONS



Next Steps

Exploring 3D modeling and going into secondary/tertiary structures. This project's demonstration was limited to 2D lattice modeling with individual amino acid sequences.

Drug Discovery Applications

As quantum-based models for the prediction of proteins' structures become more enhanced, this will enable for the synthesizing of new protein-based drugs to treat diseases.

The shape of the protein accompanied by its folding process is able to dictate its specific function in the body. By being able to predict a protein's structure, new computational drug designs can be created based on the structure of target proteins.

APPENDIX

WORKFLOW PROCESS

Preprocessing

Defining variable to hold combinations of amino acids in the chain for interaction energy computations

Defining possible directions for the random walks

Defining energy evaluations: measuring and outputting current energies, evaluating probabilities with neighboring states

```
def preprocessing_trajectory(prot_seq, sched, jobid=0):
    Nit = len(sched)
    seed = dt.now().microsecond * (jobid + 1)
    np.random.seed(seed)
    # Sequence translated to index
    s = [d[a] for a in prot_seq
    # Sequence Length
    N = len(s)
    pos = np.zeros((N, 2))
    pos[:,0] = range(N)
    tra = np.empty((Nit, N, 2))
    comb = get_combinations(pos)
    # Selecting Residue
    I = np.random.randint(2, N, size=Nit)
```

```
    # Selecting walking direction
    J = np.random.randint(3, size=Nit)
    # Third amino acid walks only up or right.
    # to avoid redundant conformations
    J[I == 2] = 0
    # Throw a dice for proposal acceptance/rejection
    K = np.random.uniform(size=Nit)
    # History of accepted conformations
    H = np.empty((Nit,))
    # Energy trace
    E = np.empty((Nit,))
    # Current energy
    E0 = total_energy(pos, s, comb)
    # main loop
    for it, (i, j, dice, T) in enumerate(zip(I, J, K, sched)):
```

WORKFLOW PROCESS

Simulated Annealing

Define sequence to fold:
YYCPETGTWYAGT

Define annealing schedule

Run simulated annealing algorithm

Total number of lowest energy conformations outputted

```
        # Delta E: change of energy will determine the probability
        # of accepting this step
        dE = E1 - E0
        # Simulated Annealing
        prob = np.exp(-dE/T)
        accept = prob > dice

        if not accept:
            pos[i:] = curr
            H[it] = 0
        else:
            E0 = E1
            H[it] = 1

        E[it] = E0
        tra[it] = pos

    return tra, H, E, seed, pos
```

```
# Current energy
    E0 = total_energy(pos, s, comb)
    for it, (i, j, dice, T) in enumerate(zip(I, J, K, sched)):

        # Store current position
        # Rrestored if walk is not successful
        curr = pos[i:].copy()

        # Using previous amino acid as reference
        back = tuple(pos[i] - pos[i-1])

        # Random walk represntation
        wdir = dirs[back][j] - back
        pos[i:] += wdir
        # New energy of the system
        E1 = total_energy(pos, s, comb)
```

WORKFLOW PROCESS

Quantum Annealing + Turn Ancilla Encoding

Define sequence to fold: YYCPETGTWYAGT

Define annealing schedule

Define energy functions: $E_{overlap}$, E_{back}, E_{pair} and adjust walk

Run quantum annealing algorithm

Total number of lowest energy conformations outputted

```python
def turn_ancilla(prot_seq, return_Hs=False):

    # length of the sequence
    N = len(prot_seq)
    s = [d[a] for a in prot_seq]
    ftq = 0
    ntq = 2 * (N-1)
    # ancilla qubits for overlap
    foq = ftq + ntq
    noq = sum([sum([u(i,j) for j in range(i+4, N+1)])
               for i in range(N-4)])
    # ancilla qubits for pairwise interactions
    fiq = foq + noq
    niq = int((N-3)*(N-2)/2)
    Nqbits = ntq + noq + niq
```

```python
# Energy Functions:
def E_back():
    l_back = 10
    return sum([back(j) for j in range(N-2)]) * l_back
def E_overlap():
    E_olap = 10
    return sum([sum([l_olap*((1+i-j)%2)*(2**u(i,j) - g(i,j) - alpha(i, j))**2
                     for j in range(i+4, N)]) for i in range(N-4)])
def E_pair():
    # eqn 32
    return sum([sum([w(i,j)*J(i,j)*(2-g(i,j)) for j in range(i+3, N)])
                for i in range(N-3)])

# Build energy terms (symbolic expressions)
Hb = (H_back())
Ho = (H_olap()) if noq > 0 else 0
Hi = (H_inte())
energy_expr = Hb + Ho + Hi
if return_Hs:
    return prot_seq, q, energy_expr, (Hb, Ho, Hi)
else:
    return prot_seq, q, energy_expr
```

WORKFLOW PROCESS

Quantum Annealing on D'Wave's Hardware

Define sequence to fold: YYCPETGTWYAGT

Define annealing schedule

Import library packages: D'Wave Sampler, Embedding Composites

Integrate with D'Wave's 2000-Q Annealer API with local machine

Define solver and corresponding adjacent graph for embedding

Adapt qubits into energy functions and scale energy landscape

```python
from dwave.system.samplers import DWaveSampler
from dwave.system.composites import EmbeddingComposite, FixedEmbeddingComposite
import dwave_networkx as dnx
from minorminer import find_embedding
import dimod
from sympy.parsing.sympy_parser import parse_expr

# Define the solver and get its correspondiing adjacency graph for embedding
solver = DWaveSampler(solver="DW_2000Q_6")
solver_G = nx.Graph(solver.edgelist)
print("Maximum anneal schedule points: {}".format(solver.properties
                                                  ["max_anneal_schedule_points"]))

# Finding energy of optimal solution
bst_00 = {}
for j, qj in enumerate(q):
    bst_00[qj] = results[best, j]
```

```python
sample_00 = dict(zip(list(bqm_00.variables),
                     [None]*len(bqm_00.variables)))
for key in bqm_00.variables:
    if type(key) == str:
        k1, k2 = parse_expr(key).as_coeff_mul()[1]
        sample_00[key] = bst_00[k1] * bst_00[k2]
    else:
        sample_00[key] = bst_00[key]

sample_spin_00 = sample_00.copy()
for key, val in sample_00.items():
    smpl_spin_00[key] = 2*val -1

mene_00 = bqm_00.energy(sample_00)
mene_spin_00 = bqm_spin_00.energy(sample_spin_00)
print((mene_00, mene_spin_00))
```

BIBLIOGRAPHY

The basic framework/steps for the simulated annealing and quantum annealing algorithms were leveraged and slightly modified from recent publications in the field (listed bellow). The construction of the energy functions, annealing schedule, and integration with D'Wave's Annealer was completed independently.

The approach presented was adapted from the following sources:

1. Dimitris Bertsimas, John Tsitsiklis. "Simulated Annealing." Statist. Sci. 8 (1) 10 - 15, February, 1993. https://doi.org/10.1214/ss/1177011077
2. Perdomo-Ortiz, A., Dickson, N., Drew-Brook, M. *et al.* Finding low-energy conformations of lattice protein models by quantum annealing. *Sci Rep* **2,** 571 (2012). https://doi.org/10.1038/srep00571
3. Robert, A., Barkoutsos, P.K., Woerner, S. *et al.* Resource-efficient quantum algorithm for protein folding. *npj Quantum Inf* **7,** 38 (2021). https://doi.org/10.1038/s41534-021-00368-4
4. Dill, K. A., Ozkan, S. B., Shell, M. S., & Weikl, T. R. (2008). The protein folding problem. *Annual review of biophysics*, *37*, 289–316. https://doi.org/10.1146/annurev.biophys.37.092707.153558
5. Chaudhuri TK, Paul S. Protein-misfolding diseases and chaperone-based therapeutic approaches. FEBS J. 2006 Apr;273(7):1331-49. doi: 10.1111/j.1742-4658.2006.05181.x. PMID: 16689923.
6. Outeiral, Carlos, et al. "The Prospects of Quantum Computing in Computational Molecular Biology." *ArXiv.org*, 26 May 2020, arxiv.org/abs/2005.12792.
7. Babej, Tomáš, et al. "Coarse-Grained Lattice Protein Folding on a Quantum Annealer." *ArXiv.org*, 2 Nov. 2018, arxiv.org/abs/1811.00713.
8. Babbush, Ryan, et al. "Construction of Energy Functions for Lattice Heteropolymer Models: A Case Study in Constraint Satisfaction Programming and Adiabatic Quantum Optimization." *ArXiv.org*, 11 June 2013, arxiv.org/abs/1211.3422.

SUPPLEMENTAL WORK

Full Code Implementation: https://github.com/aliceliu7/quantumfold

Background - Protein Folding and Drug Discovery:
https://medium.com/@aliceliu2004/protein-folding-and-drug-discovery-a-quantum-approach-6a2b08568c3a

Implementation - Quantum Annealing with 2D Lattice Protein Folding:
https://medium.com/@aliceliu2004/quantum-annealing-2d-lattice-protein-folding-31e7049aa441

CONTACT

Email: aliceliu2004@gmail.com