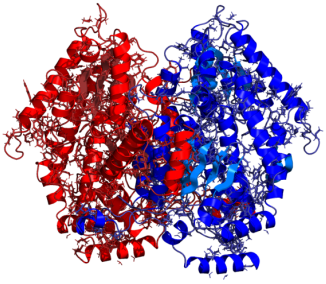
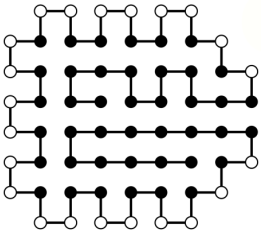
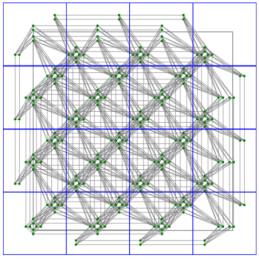






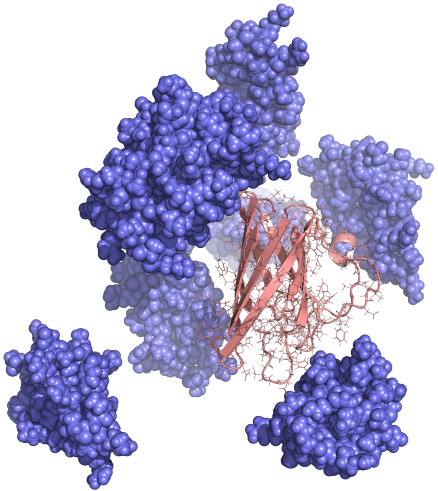
Protein folding in a 2D HP lattice model using quantum annealing

15 November 2022 | Sandipan Mohanty | Forschungszentrum Jülich, Germany

Lund University

JÜLICH Forschungszentrum

What we (SDL Biology) usually do...



- Classical Monte Carlo simulations (replica exchange, simulated tempering, simulated annealing, multi-canonical and Wang-Landau methods...)
- Atomically detailed protein model, fixed bond lengths and bond angles
- Implicit solvent interaction potential
- Markov Chain Monte Carlo simulations using torsion angles and rigid body translations/rotations as degrees of freedom
- Study phenomena involving large scale structural transitions like protein folding, aggregation, and effects of molecular crowding
- Highly optimised HPC code (ProFASi) to handle thousands of atoms

Quantenphysik

Der optimale Computer

23. Februar 2022, 11:14 Uhr | Lesezeit: 6 min

Ein Name wie ein Versprechen: „Advantage" heißt der Rechner, der so groß ist wie manches Badezimmer. (Foto: Sascha Kreklau / Forschungszentrum Jülich)

In Jülich geht der größte Quantencomputer Europas an den Start. Er soll die bestmöglichen Lösungen für komplexe mathematische Probleme finden - am Flughafen, im Pflanzenbeet oder im Aktienportfolio. Doch manchmal patzt die futuristische Technik.

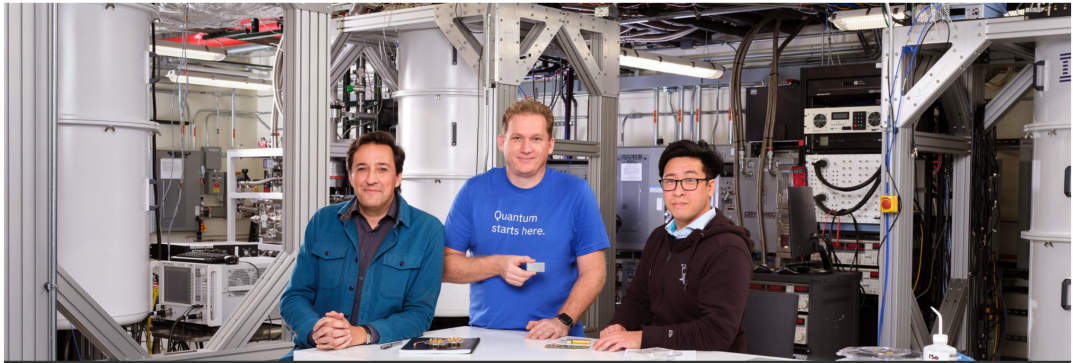
Von Christian Meier

Das neue Gebäude, das den größten Quantencomputer Europas beherbergt, sieht nicht gerade imposant aus. Hätte es Fenster in seiner holzverkleideten Außenwand, könnte es auch als

IBM Unveils 400 Qubit-Plus Quantum Processor and Next-Generation IBM Quantum System Two

Company Outlines Path Towards Quantum-Centric Supercomputing with New Hardware, Software, and System Breakthrough

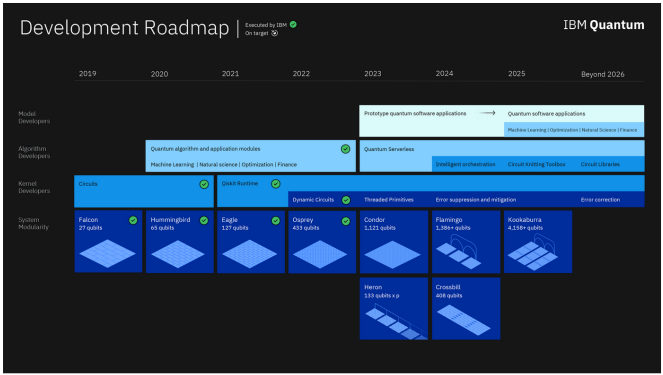
Nov 9, 2022



Dario Gil, Jay Gambetta and Jerry Chow holding the new 433 qubit 'IBM Osprey' processor

NEW YORK, Nov. 9, 2022 /PRNewswire/ -- IBM (NYSE: IBM) today kicked off the IBM Quantum Summit 2022, announcing new breakthrough advancements in quantum hardware and software and outlining its pioneering vision for quantum-centric

More Articles



At the Summit, the company unveiled the following new developments:

- **'IBM Osprey' - IBM's new 433-quantum bit (qubit) processor**

  IBM Osprey has the largest qubit count of any IBM quantum processor, more than tripling the 127 qubits on the IBM Eagle processor unveiled in 2021. This processor has the potential to run complex quantum computations well beyond the computational capability of any classical computer. For reference, the number of classical bits that would be necessary to represent a state on the IBM Osprey processor far exceeds the total number of atoms in the known universe. For more about how IBM continues to improve the scale, quality, and speed of its quantum systems, read Quantum-Centric Supercomputing: Bringing the Next Wave of Computing to Life.

- **New quantum software addresses error correction and mitigation**

  Addressing noise in quantum computers continues to be an important factor in adoption of this technology. To simplify this, IBM released a beta update to Qiskit Runtime, which now includes allowing a user to trade speed for reduced error

Crédit Mutuel Alliance Fédérale Launches Quantum Computing Readiness Discovery Phase with IBM to Establish Quantum capability in France

Additional Assets

IBM_Quantum_Flexi_Cables.jpg

IBM_Quantum_Processors.png

IBM_Quantum_System_Two_(front).jpg

Quantum computing

- Universal gate-based quantum computing
- Special-purpose quantum annealing (QA)

Quantum computing

- Universal gate-based quantum computing
- Special-purpose quantum annealing (QA)

D-Wave QA

- Minimize $E = \sum_i h_i \sigma_i + \sum_{i<j} J_{ij} \sigma_i \sigma_j$, $\sigma_i = \pm 1$
- Consider $H(t) = a(t)H_a + b(t)H_b$, $H_b \leftrightarrow E$
  $t = 0$: ground state of $H(0) \approx H_a$, $a(0) \gg b(0)$
  a and b change slowly as t increases from 0 to τ
  $t = \tau$: ground state of $H(\tau) \approx H_b$, $a(\tau) \ll b(\tau)$
  annealing rate τ^{-1} $\leftrightarrow$ cooling rate in simulated annealing
- Threats: non-adiabaticity, thermal noise, imperfect H implementation

How well does D-Wave QA work?

B. Heim, T.F. Rønnow, S.V. Isakov, M. Troyer, Quantum versus classical annealing of Ising spin glasses, *Science* **348**, 215 (2015)

"... the evidence for quantum speed-up is contradictory."


COMPUTER SCIENCE

Quantum or not, controversial computer yields no speedup

Conventional computer ties D-Wave machine

A.D. King *et al.* (D-Wave, Google), Scaling advantage over path-integral Monte Carlo in quantum simulation of geometrically frustrated magnets, *Nature Communications* **12**, 1113 (2021)

"... million-fold speedup over an efficient CPU implementation."

D-Wave quantum annealer

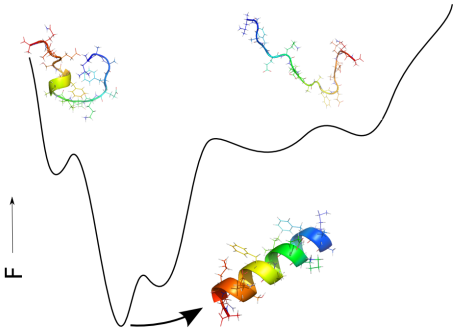
- "Frame your problem as an energy minimisation problem and solve it using the physics of the QPU, exploiting quantum fluctuations, tunneling..."

D-Wave quantum annealer

- "Frame your problem as an energy minimisation problem and solve it using the physics of the QPU, exploiting quantum fluctuations, tunneling..."
- Challenging for many computer science problems.

D-Wave quantum annealer

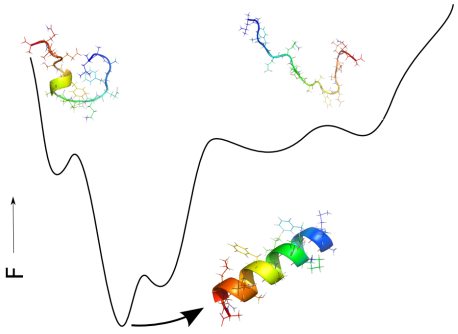
- "Frame your problem as an energy minimisation problem and solve it using the physics of the QPU, exploiting quantum fluctuations, tunneling..."
- Challenging for many computer science problems.
- But, many problems in science and engineering involve some kind of optimisation



D-Wave quantum annealer

- "Frame your problem as an energy minimisation problem and solve it using the physics of the QPU, exploiting quantum fluctuations, tunneling..."
- Challenging for many computer science problems.
- But, many problems in science and engineering involve some kind of optimisation
- What the annealer does: generate sample configurations $\{x_i\}$ of bits (equivalently, spins) which minimize the Ising model energy, or the QUBO energy

$$E = \sum_{i=1}^{N} q_i x_i + \sum_{i<j}^{N} q_{ij} x_i x_j$$

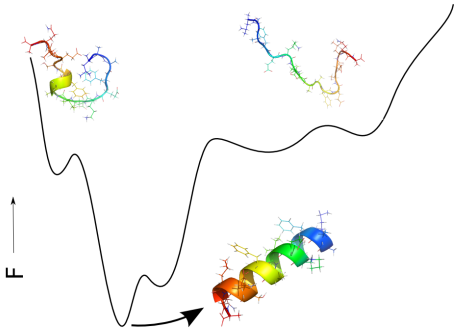


D-Wave quantum annealer

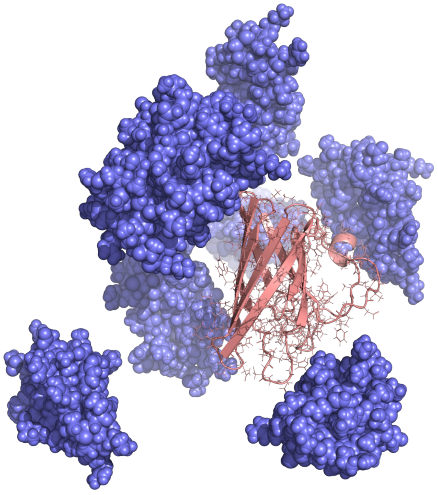
- "Frame your problem as an energy minimisation problem and solve it using the physics of the QPU, exploiting quantum fluctuations, tunneling..."
- Challenging for many computer science problems.
- But, many problems in science and engineering involve some kind of optimisation
- What the annealer does: generate sample configurations $\{x_i\}$ of bits (equivalently, spins) which minimize the Ising model energy, or the QUBO energy

$$E = \sum_{i=1}^{N} q_i x_i + \sum_{i<j}^{N} q_{ij} x_i x_j$$

- Formulating a problem for the quantum annealer: specify q_i and q_{ij} so that $\{x_i\}$ which minimizes the above energy also represents a solution



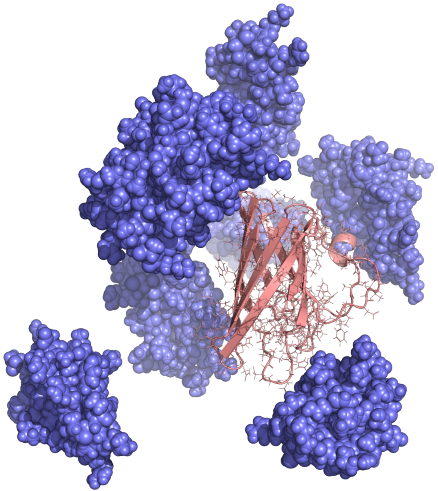
The challenge ahead...



Example study with classical MC (ProFASi)

- 9373 atoms, 2706 torsional degrees of freedom + translation/rotation of the molecules

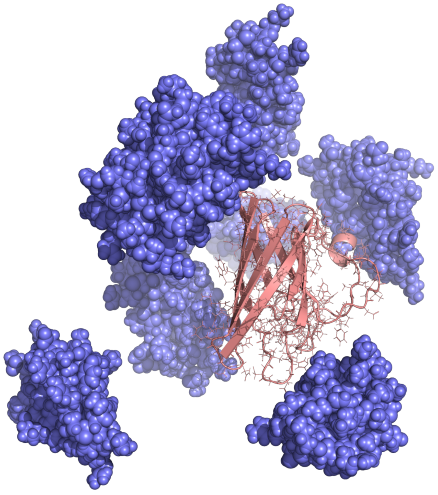
The challenge ahead...



Example study with classical MC (ProFASi)

- 9373 atoms, 2706 torsional degrees of freedom + translation/rotation of the molecules
- Because of constraints of the problem, sampling was restricted to a subset of 1783 degrees of freedom

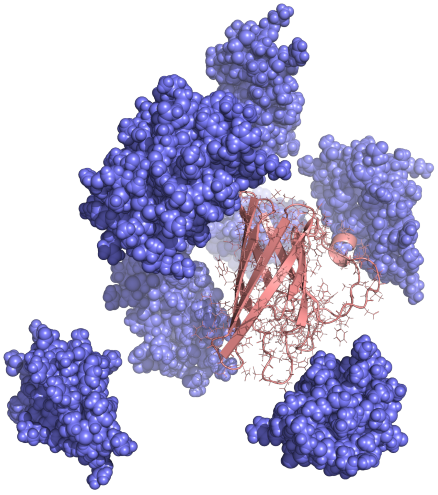
The challenge ahead...



Example study with classical MC (ProFASi)

- 9373 atoms, 2706 torsional degrees of freedom + translation/rotation of the molecules
- Because of constraints of the problem, sampling was restricted to a subset of 1783 degrees of freedom
- Vectorisation, parallelisation ... $\Longrightarrow$ $0.1ms$ per MC update

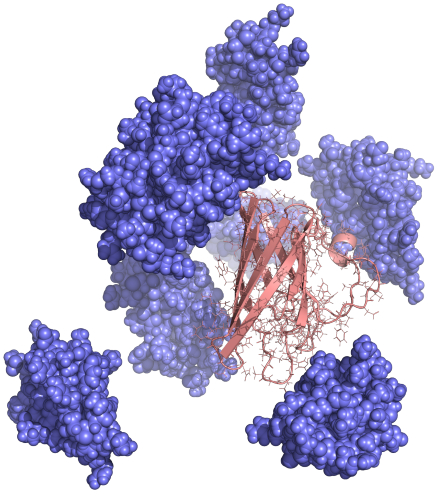
The challenge ahead...



Example study with classical MC (ProFASi)

- 9373 atoms, 2706 torsional degrees of freedom + translation/rotation of the molecules
- Because of constraints of the problem, sampling was restricted to a subset of 1783 degrees of freedom
- Vectorisation, parallelisation ... $\Longrightarrow$ $0.1ms$ per MC update
- 1.12×10^{12} MC updates were necessary to sample low energy states adequately

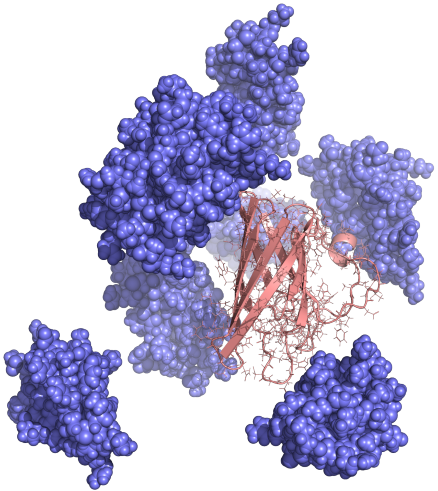
The challenge ahead...



Example study with classical MC (ProFASi)

- 9373 atoms, 2706 torsional degrees of freedom + translation/rotation of the molecules
- Because of constraints of the problem, sampling was restricted to a subset of 1783 degrees of freedom
- Vectorisation, parallelisation ... $\Longrightarrow$ $0.1ms$ per MC update
- 1.12×10^{12} MC updates were necessary to sample low energy states adequately
- Can a quantum computer perform the state space search faster?

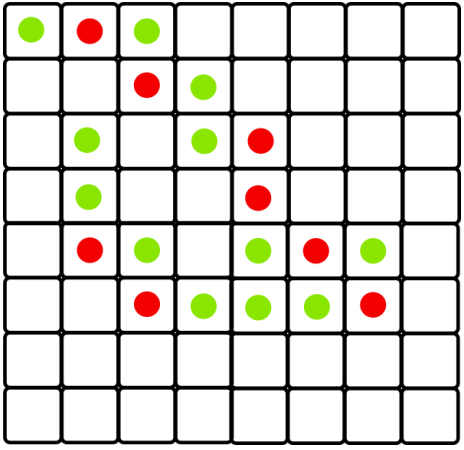
The challenge ahead...



Example study with classical MC (ProFASi)

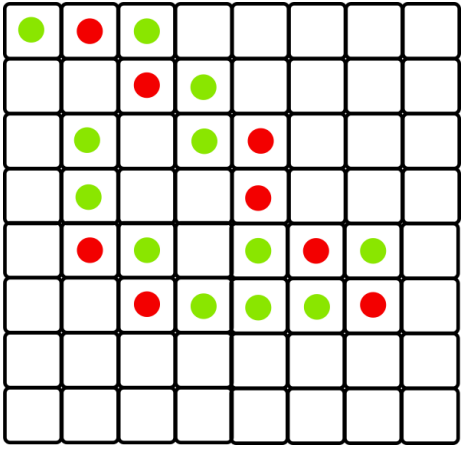
- 9373 atoms, 2706 torsional degrees of freedom + translation/rotation of the molecules
- Because of constraints of the problem, sampling was restricted to a subset of 1783 degrees of freedom
- Vectorisation, parallelisation ... $\Longrightarrow$ $0.1ms$ per MC update
- 1.12×10^{12} MC updates were necessary to sample low energy states adequately
- Can a quantum computer perform the state space search faster?
- What is our path to perform comparable studies on quantum computers?

Let's start with something simple



- One bead represents one amino acid
- HP model: Only two kinds of amino acids: H = hydrophobic, P = polar. Molecule specified by a "sequence" of H and P beads, e.g., HPHHPHPHHP
- 2D
- Beads can only be placed on lattice sites
- Two beads are said to be in contact if they occupy nearest neighbour sites on the lattice, despite not being sequence neighbours
- Attractive interaction between two H beads in contact
- Known (from classical MC studies) to capture some basic aspects of proteins

First step: a very reduced model

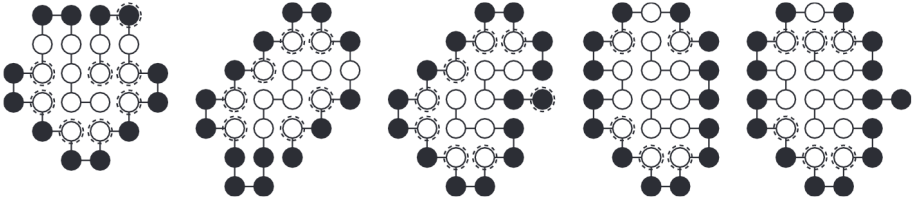


Explicit-chain MC simulations

- Sequence $S_1 S_2 S_3 \ldots$, with $S_i = H$ or P.
- $E = \sum_{i=1}^{N} \sum_{j=i+1}^{N} n(\vec{r}_i, \vec{r}_j) \epsilon_{ij}$.
- $\epsilon_{ij} = -1$ if $S_i = S_j = H$, $\epsilon_{ij} = 0$ otherwise.
- $n(\vec{r}_i, \vec{r}_j) = 1$ if $\vec{r}_i$ and $\vec{r}_j$ are nearest neighbours, $n(\vec{r}_i, \vec{r}_j) = 0$ otherwise
- Internal coordinates can be chosen as the directions of the displacements $S_1 \to S_2$, $S_2 \to S_3$, e.g., 00 =down, 01 =right, 10 =left, 11 =up
- Bit-string $\to$ conformation
- Conformation $\to$ energy
- Energy, bit flip moves $\to$ MCMC
- Works. Easy with classical MC. But that's not the topic for today!

2D HP lattice proteins

- All HP sequences with unique ground states and the corresponding structures are known[1] for $N \leq 30$
- $N = 30$: # sequences: $\sim 10^9$, # structures: $\sim 8 \times 10^{11}$



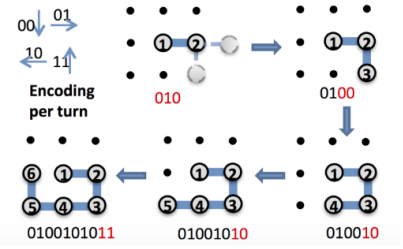
Open circle=H, filled circle=P, dashes=conserved position

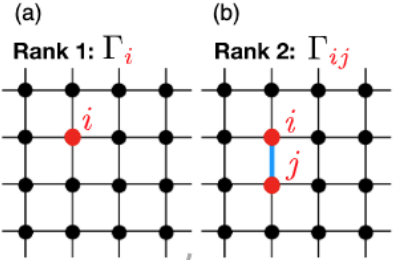
[1] A. Irbäck, C. Troein, *J. Biol. Phys.* **28**, 1 (2002); C. Holzgräfe, A. Irbäck, C. Troein *J. Chem. Phys.* **135**, 195101 (2011).

Polymer studies with quantum annealing

Short lattice proteins:
A. Perdomo-Ortiz, N. Dickson, M. Drew-Brook, G. Rose, A. Aspuru-Guzik, Finding low-energy conformations of lattice protein models by quantum annealing, *Sci. Rep.* **2**, 248 (2012)

Homopolymers:
C. Micheletti, P. Hauke, P. Faccioli, Polymer physics by quantum computing, *Phys. Rev. Lett.* **127**, 080501 (2021)





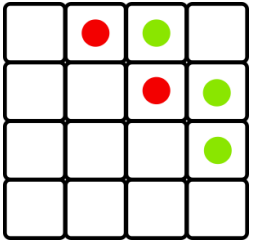
Polymer studies with quantum annealing

Some difficulties encountered in previous attempts...

- Growth algorithms: hard to encode self-avoidance
- Qubit-economic encoding of chain conformations, but needed extra qubits to ensure a quadratic Hamiltonian
- Non-trivial to add more beads to the studied polymer

Spin glass representation of HP chains: our formulation

- Chain with N beads on square grid with L^2 sites
- $\sigma_s^f = 1$ if chain bead f is on lattice site s, $\sigma_s^f = 0$ otherwise
- $N \times L^2$ binary variables σ_s^f
- Not all possible $\{\sigma_s^f\}$ represent valid chains. Valid configurations emerge dynamically



f	σ_s^f
1	0100 0000 0000 0000
2	0010 0000 0000 0000
3	0000 0010 0000 0000
4	0000 0001 0000 0000
5	0000 0000 0001 0000

Energy function:

$$E = E_{\mathrm{HP}} + \sum_{i=1}^{3} \lambda_i E_i$$

$E_{\mathrm{HP}} =$ HP energy

$E_i =$ constraint energies (≥ 0)

$\lambda_i =$ strength parameters

Valid chain configuration $\Leftrightarrow$ $E_1 = E_2 = E_3 = 0$

Spin glass representation of HP chains

HP energy (sequence $h_1, \ldots, h_N$):

$$E_{\mathrm{HP}} = - \sum_{\substack{|f-f'|>1 \\ h_f = h_{f'} = \mathrm{H}}} \sum_{\substack{\text{n.n. pairs} \\ s,s'}} \sigma_s^f \sigma_{s'}^{f'}$$

Site allotment to beads (each bead on one and only one lattice site):

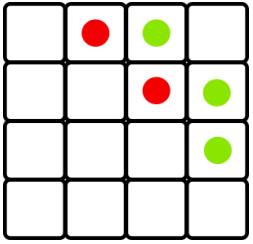
$$E_1 = \sum_{f=1}^{N} \left(\sum_s \sigma_s^f - 1 \right)^2$$

Self-avoidance:

$$E_2 = \frac{1}{2} \sum_{f \neq f'} \sum_s \sigma_s^f \sigma_s^{f'}$$

Chain connectivity:

$$E_3 = \sum_{f=1}^{N-1} \sum_s \sigma_s^f \sum_{||s'-s||>1} \sigma_{s'}^{f+1}$$

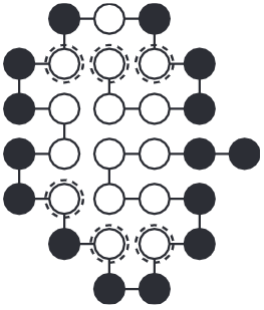


f	σ_s^f
1	0100 0000 0000 0000
2	0010 0000 0000 0000
3	0000 0010 0000 0000
4	0000 0001 0000 0000
5	0000 0000 0001 0000

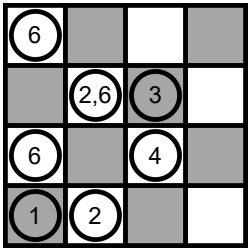
Test: minimize E by classical simulated annealing (SA)

$N = 30$

- 10^2 grid, 30×10^2 binary variables, $2^{3000} \sim 10^{903}$ states
- SA with 25 temperature levels, 3 core minutes per SA run
- Success rate $\gtrsim 10\%$
- May compare with conventional explicit-chain SA:
  8×10^{11} states, 10 core seconds per SA run, success rate $>50\%$



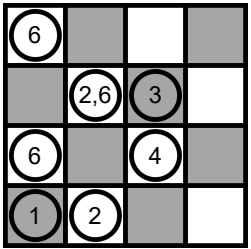
Binary field approach, classical SA, typical run



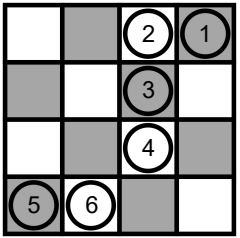
(a)

- Random start: all constraints likely violated

Binary field approach, classical SA, typical run



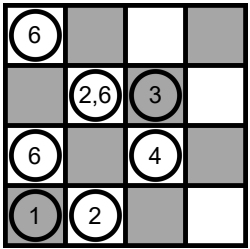
(a)



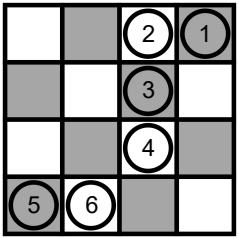
(b)

- Random start: all constraints likely violated
- Gradually, lower energy states are found with more constraints satisfied

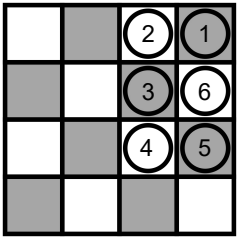
Binary field approach, classical SA, typical run



(a)



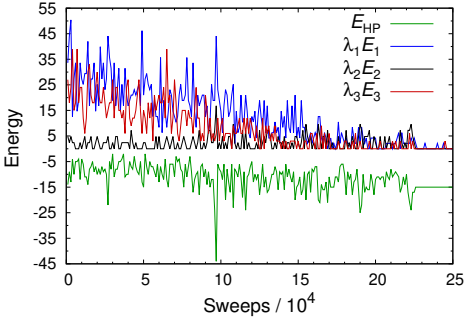
(b)

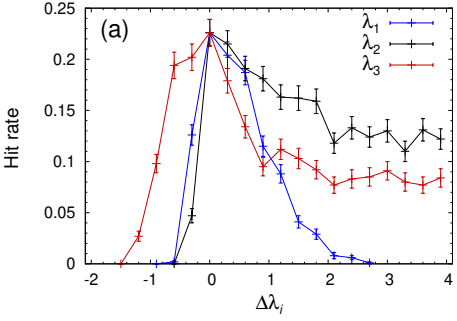


(c)

- Random start: all constraints likely violated
- Gradually, lower energy states are found with more constraints satisfied
- An annealing run, may end in a low energy state where all constraints are satisfied and with a low HP energy

Binary field approach, typical run, parameter sensitivity





- Constraint terms decay at different rates
- The main physical term might go lower than its optimal value during the run when violating constraints

- If constraints are weighted too low, they don't relax fast enough. If they are too strong the energy landscape is rugged and the sampling is slow.
- Some crude tuning required to get good success rates.

Using the D-Wave hybrid solver

- Binary variables: σ_s^f, where s is a lattice position (x, y) and f is the serial number of a given bead along the sequence
- To formulate our QBM (quadratic binary model), we need to map the total contribution involving a pair of these variables to a pairwise contribution
- The public interface expects Q to be given as a `dictionary`
  - **key type:** An ordered pair ($k1$, $k2$), where $k1$, $k2$ are keys identifying two of our binary variables. For instance, if we identify our σ_s^f variables in terms of the x, y lattice coordinates and the serial number f along the sequence, the key type for Q would be $(((x_1, y_1), f_1), ((x_2, y_2), f_2))$.
  - **value type:** numeric

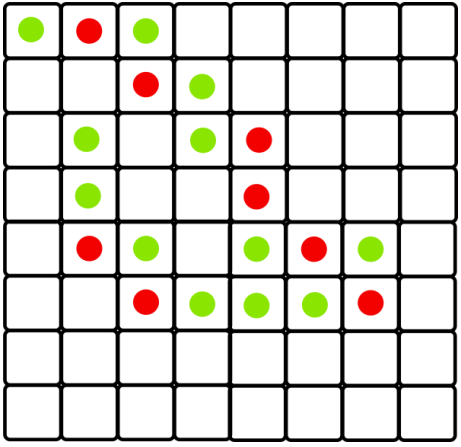
$$E_{\mathrm{HP}} = -\sum_{\substack{|f-f'|>1 \\ h_f = h_{f'} = \mathrm{H}}} \sum_{\substack{\text{n.n. pairs} \\ s,s'}} \sigma_s^f \sigma_{s'}^{f'}$$

$$E_1 = \sum_{f=1}^{N} \left(\sum_s \sigma_s^f - 1 \right)^2$$

$$E_2 = \frac{1}{2} \sum_{f \neq f'} \sum_s \sigma_s^f \sigma_s^{f'}$$

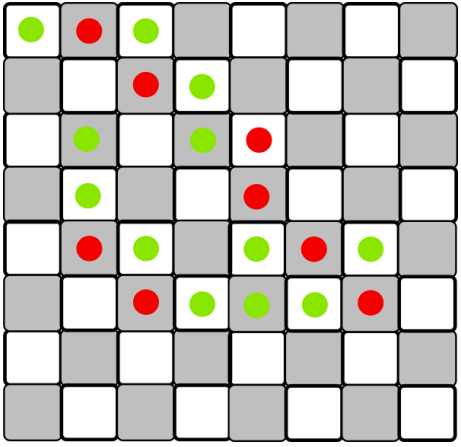
$$E_3 = \sum_{f=1}^{N-1} \sum_s \sigma_s^f \sum_{||s'-s||>1} \sigma_{s'}^{f+1}$$

A trick to reduce the number of required qbits



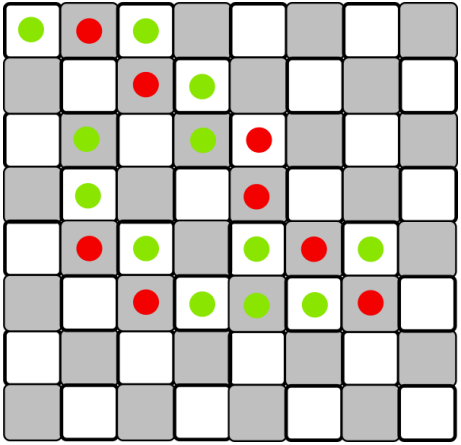
- **Why:** As presented up to this point, our QBM needs (for sequence length N on an $L \times L$ lattice) $N \times L^2$ bits in its state representation, i.e., 3000 bits for $N = 30$, $L = 10$

A trick to reduce the number of required qbits



- **Why:** As presented up to this point, our QBM needs (for sequence length N on an $L \times L$ lattice) $N \times L^2$ bits in its state representation, i.e., 3000 bits for $N = 30$, $L = 10$
- *Trick:* Colour lattice sites like a chess board. If the sequence position 1 is on a white site, positions $3, 5, 7 \ldots$ are also on white sites! And similarly for the black lattice sites.

A trick to reduce the number of required qbits



- **Why:** As presented up to this point, our QBM needs (for sequence length N on an $L \times L$ lattice) $N \times L^2$ bits in its state representation, i.e., 3000 bits for $N = 30$, $L = 10$
- *Trick:* Colour lattice sites like a chess board. If the sequence position 1 is on a white site, positions $3, 5, 7 \ldots$ are also on white sites! And similarly for the black lattice sites.
- Every odd sequence position has $\approx \frac{1}{2}L^2$ available sites, and similarly for even sequence positions. Number of possible (s, f) combinations for σ_s^f

$$\approx \frac{1}{2}N \times \frac{1}{2}L^2 + \frac{1}{2}N \times \frac{1}{2}L^2 = \frac{1}{2}N \times L^2$$

Useful bits from the code

```
1  def parity(node):
2      return (node[0]%2 + node[1]%2)%2
3  def E_HP_qubo_contribs(g, sequence): # g is a graph representing the L x L lattice
4      QQ = defaultdict(float)
5      for u, v in g.edges():
6          if parity(u) == 0:
7              ev, od = u, v
8          else:
9              ev, od = v, u
10         for i in range(len(sequence)):
11             for j in range(i+1, len(sequence)):
12                 if (i - j) * (i - j) <= 4:
13                     continue
14                 if (i + j) % 2 == 0:
15                     continue
16                 if (sequence[i] == 0 or sequence[j] == 0):
17                     continue;
18                 if (i % 2 == 0):
19                     QQ[((ev, i), (od, j))] += -1
20                 else:
21                     QQ[((ev, j), (od, i))] += -1
22
23     return QQ
```

Useful bits from the code

```
def unique_bead_location(g, strength, sequence_length):
    evenpos = [i for i in range(sequence_length) if i%2 == 0]
    oddpos = [i for i in range(sequence_length) if i%2 == 1]
    QQ = defaultdict(float)
    for i in evenpos:
        for u in g.nodes():
            if parity(u) == 0:
                QQ[((u, i), (u, i))] += -strength
        for u in g.nodes():
            for v in g.nodes():
                if u != v and parity(u) == 0 and parity(v) == 0:
                    QQ[((u, i), (v, i))] += strength
    for i in oddpos:
        for u in g.nodes():
            if parity(u) == 1:
                QQ[((u, i), (u, i))] += -strength
        for u in g.nodes():
            for v in g.nodes():
                if u != v and parity(u) == 1 and parity(v) == 1:
                    QQ[((u, i), (v, i))] += strength
    return QQ
```

Useful bits from the code

```
1
2  def self_avoidance(g, strength, sequence_length):
3      evenpos = [i for i in range(sequence_length) if i%2 == 0]
4      oddpos = [i for i in range(sequence_length) if i%2 == 1]
5      QQ = defaultdict(float)
6      for u in g.nodes():
7          if parity(u) == 0:
8              for x in evenpos:
9                  for y in evenpos:
10                     if x < y:
11                         QQ[((u, x), (u, y))] += strength
12         else:
13             for x in oddpos:
14                 for y in oddpos:
15                     if x < y:
16                         QQ[((u, x), (u, y))] += strength
17
18     return QQ
```

Note: Nothing D-Wave specific in any of these functions! Just plain old python.

Useful bits from the code

```
1
2  def chain_connectivity(g, strength, sequence_length):
3      evenpos = [i for i in range(sequence_length) if i%2 == 0]
4      oddpos = [i for i in range(sequence_length) if i%2 == 1]
5      QQ = defaultdict(float)
6      for i in evenpos:
7          for u in g.nodes():
8              for v in g.nodes():
9                  if u == v or ((u,v) in g.edges()) or ((v,u) in g.edges()):
10                     continue
11                 if parity(u) == 0 and parity(v) == 1:
12                     if i != evenpos[-1] or sequence_length % 2 == 0:
13                         QQ[((u, i), (v, i+1))] += strength
14     for i in oddpos:
15         for u in g.nodes():
16             for v in g.nodes():
17                 if u == v or ((u,v) in g.edges()) or ((v,u) in g.edges()):
18                     continue
19                 if parity(u) == 0 and parity(v) == 1:
20                     if i != oddpos[-1] or sequence_length % 2 == 1:
21                         QQ[((u, i+1), (v, i))] += strength
22     return QQ
23
```

Sampling our QBM using the hybrid solver

```
1  def lattice_HP_QUBO(dim, L, S):
2      # add up contributions and return total Q
3  sequence = from_str("HPPHPPHPH PPHPHPPH PPHHH")
4  dim = [10, 10]
5  Q = lattice_HP_QUBO(dim, Lambda, sequence)
6  # Nothing "quantum" up to this point
7  # Need to find {x_i} minimising Xt Q X
8  from dwave.system import LeapHybridSampler
9  sampler = LeapHybridSampler()
10 for i in range(0, 10):
11     sampleset = sampler.sample_qubo(Q,time_limit=8)
12     # ...
```

- Set up Q (a mapping from a pairs of qbit ids to real numbers)

Sampling our QBM using the hybrid solver

```
1  def lattice_HP_QUBO(dim, L, S):
2      # add up contributions and return total Q
3  sequence = from_str("HPPHPPHPH PPHPHPPH PPHHH")
4  dim = [10, 10]
5  Q = lattice_HP_QUBO(dim, Lambda, sequence)
6  # Nothing "quantum" up to this point
7  # Need to find {x_i} minimising Xt Q X
8  from dwave.system import LeapHybridSampler
9  sampler = LeapHybridSampler()
10 for i in range(0, 10):
11     sampleset = sampler.sample_qubo(Q,time_limit=8)
12     # ...
```

- Set up Q (a mapping from a pairs of qbit ids to real numbers)
- Create a sampler

Sampling our QBM using the hybrid solver

```
1  def lattice_HP_QUBO(dim, L, S):
2      # add up contributions and return total Q
3  sequence = from_str("HPPHPPHPH PPHPHPPH PPHHH")
4  dim = [10, 10]
5  Q = lattice_HP_QUBO(dim, Lambda, sequence)
6  # Nothing "quantum" up to this point
7  # Need to find {x_i} minimising Xt Q X
8  from dwave.system import LeapHybridSampler
9  sampler = LeapHybridSampler()
10 for i in range(0, 10):
11     sampleset = sampler.sample_qubo(Q,time_limit=8)
12     # ...
```

- Set up Q (a mapping from a pairs of qbit ids to real numbers)
- Create a sampler
- Ask the sampler to return a sample bit vector minimising the energy function encoded in Q

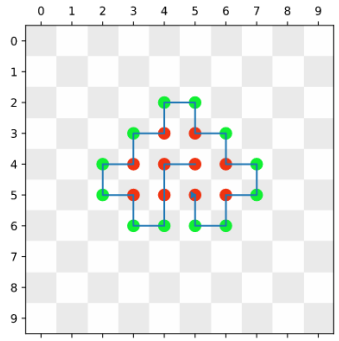
Sampling our QBM using the hybrid solver

```
1  def lattice_HP_QUBO(dim, L, S):
2      # add up contributions and return total Q
3  sequence = from_str("HPPHPPHPH PPHPHPPH PPHHH")
4  dim = [10, 10]
5  Q = lattice_HP_QUBO(dim, Lambda, sequence)
6  # Nothing "quantum" up to this point
7  # Need to find {x_i} minimising Xt Q X
8  from dwave.system import LeapHybridSampler
9  sampler = LeapHybridSampler()
10 for i in range(0, 10):
11     sampleset = sampler.sample_qubo(Q,time_limit=8)
12     # ...
```

- Set up Q (a mapping from a pairs of qbit ids to real numbers)
- Create a sampler
- Ask the sampler to return a sample bit vector minimising the energy function encoded in Q
- Continue with analysis

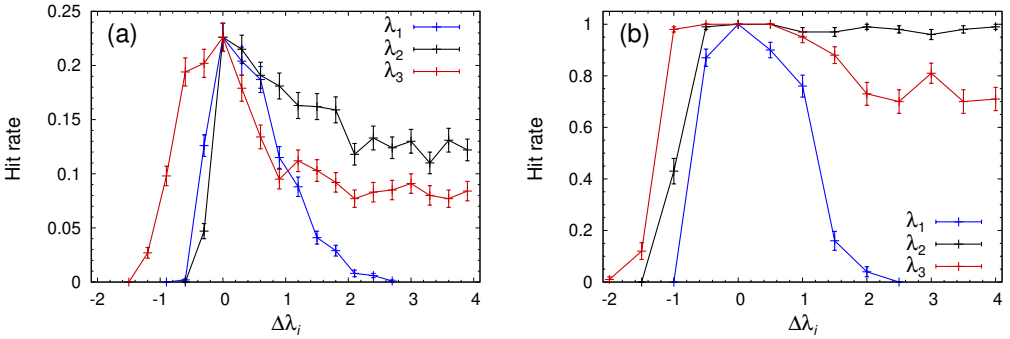
Sampling our QBM using the hybrid solver

```
1   def lattice_HP_QUBO(dim, L, S):
2       # add up contributions and return total Q
3   sequence = from_str("HPPHPPHPH PPHPHPPH PPHHH")
4   dim = [10, 10]
5   Q = lattice_HP_QUBO(dim, Lambda, sequence)
6   # Nothing "quantum" up to this point
7   # Need to find {x_i} minimising Xt Q X
8   from dwave.system import LeapHybridSampler
9   sampler = LeapHybridSampler()
10  for i in range(0, 10):
11      sampleset = sampler.sample_qubo(Q,time_limit=8)
12      # ...
```



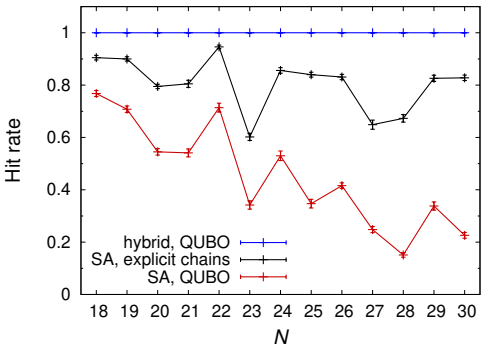
- Set up Q (a mapping from a pairs of qbit ids to real numbers)
- Create a sampler
- Ask the sampler to return a sample bit vector minimising the energy function encoded in Q
- Continue with analysis

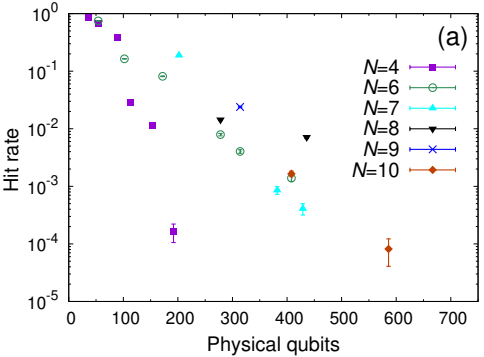
Hybrid solver: optimal λ_i parameters



- Vaguely similar shapes compared to classical SA simulations
- Maximum achievable hit rate is **far superior!**

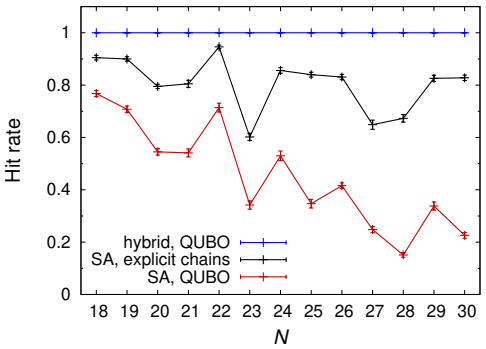
Hybrid solver: comparison with classical SA and pure QPU

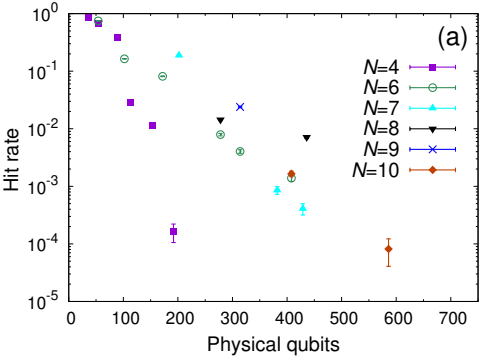




- Hybrid solver reproduces the known ground states in about 100% of the runs for sequences up to 30 beads
- Classical SA with the same encoding suffers from having a large number of DOF
- Classical SA with explicit chains is fast, results are good, but still can't keep up!

Hybrid solver: comparison with classical SA and pure QPU

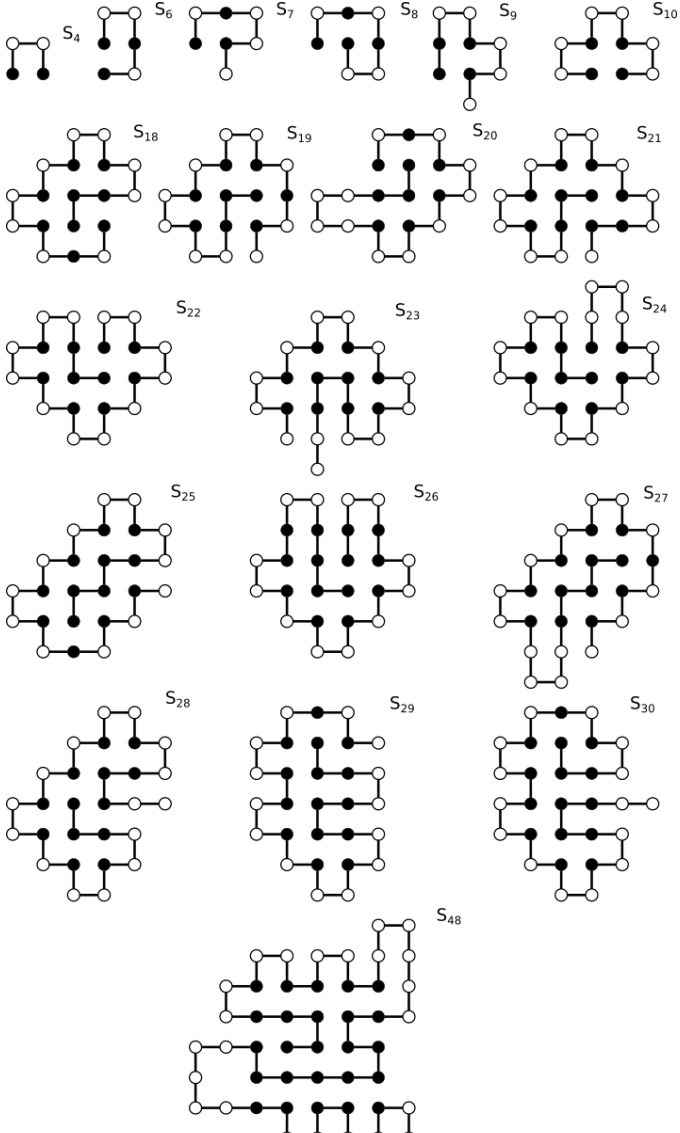




- Hybrid solver reproduces the known ground states in about 100% of the runs for sequences up to 30 beads
- Pure QPU sampler performs poorly with even 10 bead sequences
- The largest sequence we managed with pure QPU was 14 beads long

HP sequences up to 30 beads

Name	Sequence	$E_{\min}$
S_4	HPPH	-1
S_6	HPPHPH	-2
S_7	PHPPHPH	-2
S_8	HPHPHPPH	-3
S_9	HHPPHPPHP	-3
S_{10}	HPPHPPHPPH	-4
S_{14}	HHHPPPHPPHPPPH	-5
S_{18}	HHHPPHPPHPHPPHPHPH	-9
S_{19}	PHPHPHPPHPHPPHPPHHH	-9
S_{20}	HPHPHPPHPHPPHPPPHHH	-9
S_{21}	PHHPPHPHPPHPHPPHPPHHH	-10
S_{22}	HPPHPPHPHPPHPHPPHPPHHH	-11
S_{23}	PPHHHHPPHPPHPHPPHPHPPHP	-10
S_{24}	HPPPPHPPHPHPPHPHPPHPPHHH	-11
S_{25}	PHPHPHPHPPHPHPHPPHPPHHHHH	-13
S_{26}	HHHHPPHHPPHPHPPHPHPPHHPPHH	-14
S_{27}	PHPHPHPHPPHPHPHPPHPPPPHHHHH	-13
S_{28}	PPHHHPPHPPHPHPHPPHPHPPHPPHHH	-13
S_{29}	PHPHPHPPHHPPHPHPPHPPHHHHPPHHH	-15
S_{30}	PPHHHHPPHPPHPHPPHHPPHPHPHPPHHH	-15


S4
S6
S7
S8
S9
S10
S18
S19
S20
S21
S22
S23
S24
S25
S26
S27
S28
S29
S30
S48

How about longer sequences?

- For chain lengths > 30, exhaustive enumeration is not available

How about longer sequences?

- For chain lengths > 30, exhaustive enumeration is not available
- Whether or not any given sequence really has a unique ground state is not known

How about longer sequences?

- For chain lengths > 30, exhaustive enumeration is not available
- Whether or not any given sequence really has a unique ground state is not known
- Some longer sequences have been studied in detailed MC simulations

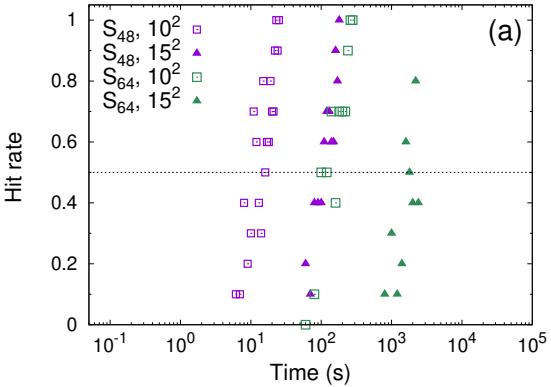
How about longer sequences?

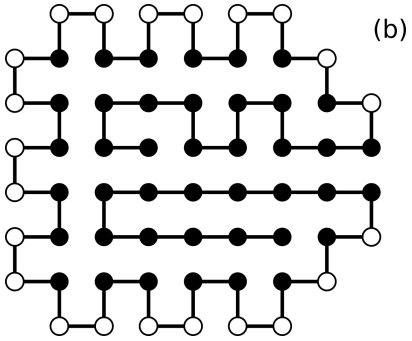
- For chain lengths > 30, exhaustive enumeration is not available
- Whether or not any given sequence really has a unique ground state is not known
- Some longer sequences have been studied in detailed MC simulations
- S48: PPHPPHHP PHHPPPPP HHHHHHHH HHPPPPPP HHPPHHPP HPPHHHHH,
  lowest known energy = -23
  F. Liang and W. H. Wong, Evolutionary Monte Carlo for protein folding simulations, *J. Chem. Phys.* **115**, 3374 (2001).

How about longer sequences?

- For chain lengths > 30, exhaustive enumeration is not available
- Whether or not any given sequence really has a unique ground state is not known
- Some longer sequences have been studied in detailed MC simulations
- S48: PPHPPHHP PHHPPPPP HHHHHHHH HHPPPPPP HHPPHHPP HPPHHHHH,
  lowest known energy = -23
  F. Liang and W. H. Wong, Evolutionary Monte Carlo for protein folding simulations, *J. Chem. Phys.* **115**, 3374 (2001).
- S64: HHHHHHHH HHHHPHPH PPHHPPHH PPHPPHHP PHHPPHPP HHPPHHPP HPHPHHHH HHHHHHHH,
  lowest known energy = -42
  U. Bastolla, H. Frauenkron, E. Gerstner, P. Grassberger, and W. Nadler, Testing a new Monte Carlo algorithm for protein folding, *Proteins* **32**, 52 (1998).

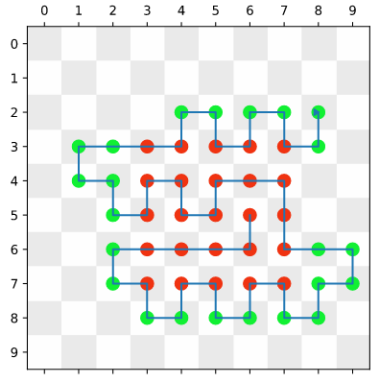
How about longer sequences?

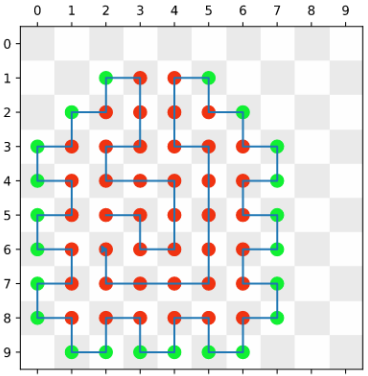




- Initial attempts: only 1 hybrid annealing run in 300 found a state of energy -42
- The hybrid annealer has a default time limit 8s for the 64 bead system. Increasing this limit beyond a threshold rapidly increased the number of annealing cycles ending in that ground state.

How about longer sequences?





- Longer run times need to be given to the hybrid annealer for longer sequences
- Slight changes in λ_i parameters (weights for constraints) also required
- With these changes, we get 100% success rate to the lowest energy state

Wish list

- More openness: How does the hybrid annealer work? What preprocessing steps does it undertake?
- Direct interface to `C`, `C++`, `FORTRAN`... (I need my overloaded functions, `const`, `constepxr`, `template`, ...)
- JSC queue system integration?
- SSH access?

Summary

- "Folding lattice proteins with quantum annealing" Anders Irbäck, Lucas Knuthson, Sandipan Mohanty, and Carsten Peterson Phys. Rev. Research 4, 043013

Summary

- "Folding lattice proteins with quantum annealing" Anders Irbäck, Lucas Knuthson, Sandipan Mohanty, and Carsten Peterson Phys. Rev. Research 4, 043013
- Continuation:

Summary

- "Folding lattice proteins with quantum annealing" Anders Irbäck, Lucas Knuthson, Sandipan Mohanty, and Carsten Peterson Phys. Rev. Research 4, 043013
- Continuation:
  - How can we get better results with pure QPU runs?

Summary

- "Folding lattice proteins with quantum annealing" Anders Irbäck, Lucas Knuthson, Sandipan Mohanty, and Carsten Peterson Phys. Rev. Research 4, 043013
- Continuation:
  - How can we get better results with pure QPU runs?
  - Different longer sequences

Summary

- "Folding lattice proteins with quantum annealing" Anders Irbäck, Lucas Knuthson, Sandipan Mohanty, and Carsten Peterson Phys. Rev. Research 4, 043013
- Continuation:
  - How can we get better results with pure QPU runs?
  - Different longer sequences
  - Better prior estimation of required annealing times

Summary

- "Folding lattice proteins with quantum annealing" Anders Irbäck, Lucas Knuthson, Sandipan Mohanty, and Carsten Peterson Phys. Rev. Research 4, 043013
- Continuation:
  - How can we get better results with pure QPU runs?
  - Different longer sequences
  - Better prior estimation of required annealing times
  - 3D

Summary

- "Folding lattice proteins with quantum annealing" Anders Irbäck, Lucas Knuthson, Sandipan Mohanty, and Carsten Peterson Phys. Rev. Research 4, 043013
- Continuation:
  - How can we get better results with pure QPU runs?
  - Different longer sequences
  - Better prior estimation of required annealing times
  - 3D
  - More amino acid categories

Summary

- "Folding lattice proteins with quantum annealing" Anders Irbäck, Lucas Knuthson, Sandipan Mohanty, and Carsten Peterson Phys. Rev. Research 4, 043013
- Continuation:
  - How can we get better results with pure QPU runs?
  - Different longer sequences
  - Better prior estimation of required annealing times
  - 3D
  - More amino acid categories
  - More detailed representation of the molecules

Summary

- "Folding lattice proteins with quantum annealing" Anders Irbäck, Lucas Knuthson, Sandipan Mohanty, and Carsten Peterson Phys. Rev. Research 4, 043013
- Continuation:
  - How can we get better results with pure QPU runs?
  - Different longer sequences
  - Better prior estimation of required annealing times
  - 3D
  - More amino acid categories
  - More detailed representation of the molecules
- Acknowledgements

Summary

- "Folding lattice proteins with quantum annealing" Anders Irbäck, Lucas Knuthson, Sandipan Mohanty, and Carsten Peterson Phys. Rev. Research 4, 043013
- Continuation:
  - How can we get better results with pure QPU runs?
  - Different longer sequences
  - Better prior estimation of required annealing times
  - 3D
  - More amino acid categories
  - More detailed representation of the molecules
- Acknowledgements
  - Active collaboration with Lund University, Sweden and Wallenberg Centre for Quantum Technology at Chalmers University, Sweden.

Summary

- "Folding lattice proteins with quantum annealing" Anders Irbäck, Lucas Knuthson, Sandipan Mohanty, and Carsten Peterson Phys. Rev. Research 4, 043013
- Continuation:
  - How can we get better results with pure QPU runs?
  - Different longer sequences
  - Better prior estimation of required annealing times
  - 3D
  - More amino acid categories
  - More detailed representation of the molecules
- Acknowledgements
  - Active collaboration with Lund University, Sweden and Wallenberg Centre for Quantum Technology at Chalmers University, Sweden.
  - Thanks to: Dennis Willsch (JSC), Kristel Michielsen (JSC), and Hanna Linn (Chalmers).