Lecture Topic: Applications in Financial Services

Practical aspects of quantum computing

Today, we want to make a few remarks on the practical aspects of use of what we have seen in the financial services industry. Therein, one needs to deal with many more vendors (i.e., salesmen) than universities (i.e., researchers). Let us start with a few remarks on the practical aspects of quantum annealers, and the vendors thereof.

D-Wave Ocean

While there are several quantum annealers across the world in academic environments, the most well-known vendor is D-Wave Systems.

You can experiment with `pip install dwave-ocean-sdk`, D-Wave's quantum annealing emulator.

Other vendors that develop superconducting quantum annealers are Qilimanjaro and Avaqus.

D-Wave Ocean

While there are several quantum annealers across the world in academic environments, the most well-known vendor is D-Wave Systems.

You can experiment with `pip install dwave-ocean-sdk`, D-Wave's quantum annealing emulator.

Other vendors that develop superconducting quantum annealers are Qilimanjaro and Avaqus.

D-Wave Ocean

While there are several quantum annealers across the world in academic environments, the most well-known vendor is D-Wave Systems.

You can experiment with `pip install dwave-ocean-sdk`, D-Wave's quantum annealing emulator.

Other vendors that develop superconducting quantum annealers are Qilimanjaro and Avaqus.

Analogue Quantum Computation

Recall, QA is a type of analog quantum computation based on the concept of adiabatic quantum computation (AQC).

As such, it is possible to devise systems that perform AQC with stoquastic Hamiltonians but are not necessarily based on superconducting qubits.

Such examples include Pasqal and QuEra that use arrays of Rydberg atoms which are highly excited atoms with a large distance between the electron and the nucleus.

Analogue Quantum Computation

Recall, QA is a type of analog quantum computation based on the concept of adiabatic quantum computation (AQC).

As such, it is possible to devise systems that perform AQC with stoquastic Hamiltonians but are not necessarily based on superconducting qubits.

Such examples include Pasqal and QuEra that use arrays of Rydberg atoms which are highly excited atoms with a large distance between the electron and the nucleus.

Analogue Quantum Computation

Recall, QA is a type of analog quantum computation based on the concept of adiabatic quantum computation (AQC).

As such, it is possible to devise systems that perform AQC with stoquastic Hamiltonians but are not necessarily based on superconducting qubits.

Such examples include Pasqal and QuEra that use arrays of Rydberg atoms which are highly excited atoms with a large distance between the electron and the nucleus.

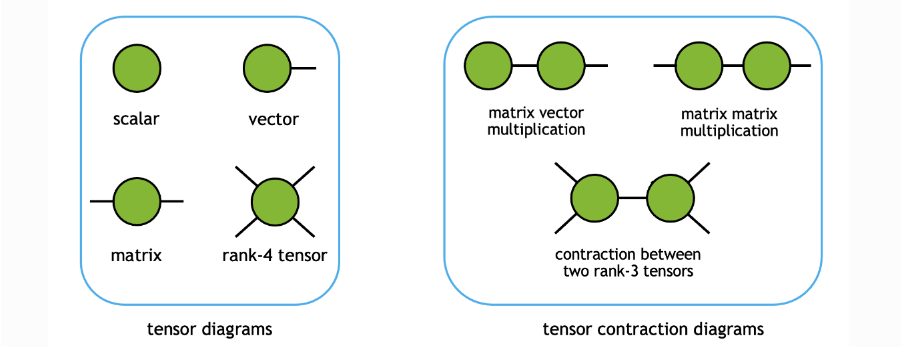
Hybrid Solvers

Several companies manufacture specialized classical hardware (e.g., based on FPGAs) that simulate quantum annealing, for example Fujitsu and Hitachi.

Tensor Networks

Finally, let's mention another approach to solve a number of interesting optimization problems, again via a QUBO reformulation, called **tensor networks**. NVIDIA is a major player.

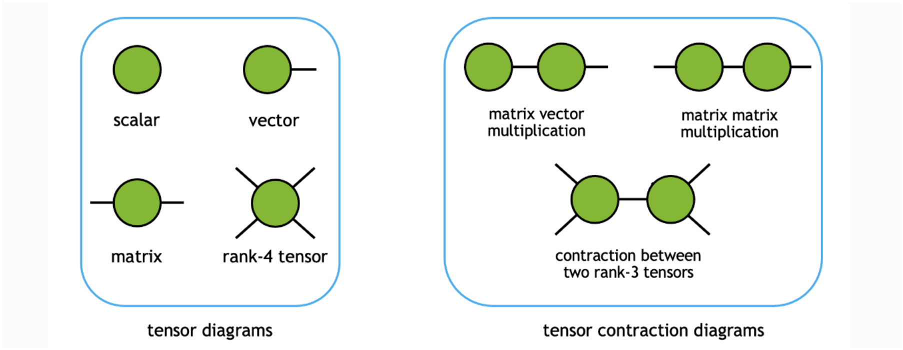
Essentially, tensor networks clasically "mimic" the behavior of a large number of **weakly entangled** quantum states.


scalar
vector
matrix
rank-4 tensor
tensor diagrams
matrix vector multiplication
matrix matrix multiplication
contraction between two rank-3 tensors
tensor contraction diagrams

Tensor Networks

Finally, let's mention another approach to solve a number of interesting optimization problems, again via a QUBO reformulation, called **tensor networks**. NVIDIA is a major player.

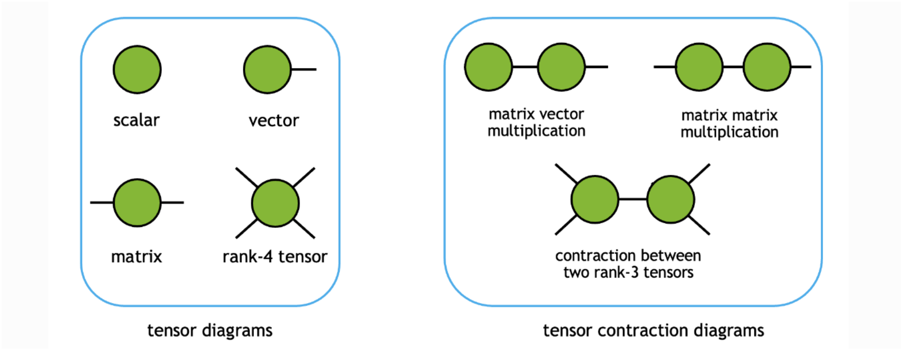
Essentially, tensor networks clasically "mimic" the behavior of a large number of **weakly entangled** quantum states.


scalar
vector
matrix
rank-4 tensor
tensor diagrams
matrix vector multiplication
matrix matrix multiplication
contraction between two rank-3 tensors
tensor contraction diagrams

Tensor Networks

Finally, let's mention another approach to solve a number of interesting optimization problems, again via a QUBO reformulation, called **tensor networks**. NVIDIA is a major player.

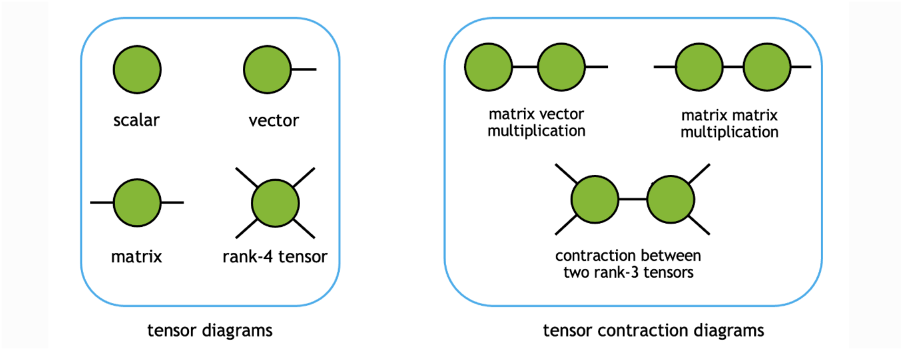
Essentially, tensor networks clasically "mimic" the behavior of a large number of **weakly entangled** quantum states.


scalar
vector
matrix
rank-4 tensor
tensor diagrams
matrix vector multiplication
matrix matrix multiplication
contraction between two rank-3 tensors
tensor contraction diagrams

Tensor Networks

Finally, let's mention another approach to solve a number of interesting optimization problems, again via a QUBO reformulation, called **tensor networks**. NVIDIA is a major player.

Essentially, tensor networks clasically "mimic" the behavior of a large number of **weakly entangled** quantum states.

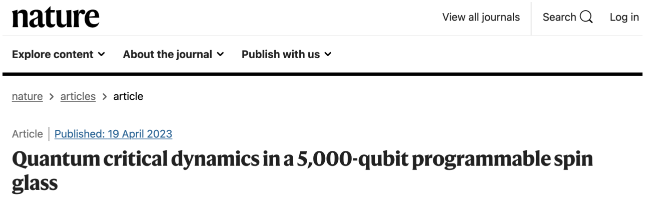

scalar
vector
matrix
rank-4 tensor
tensor diagrams
matrix vector multiplication
matrix matrix multiplication
contraction between two rank-3 tensors
tensor contraction diagrams

Focus: D-Wave

D-Wave being the first company to file for a patent.

D-Wave gained a lot of notice once multi-qubit quantum tunneling effects were observed experimental and showed the computational potential it may have.

Their most advanced is the 5,760-qubit Advantage machine with which the following study was performed:



nature

View all journals | Search | Log in

Explore content | About the journal | Publish with us

nature > articles > article

Article | Published: 19 April 2023

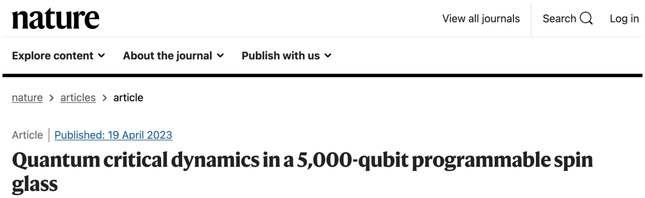
Quantum critical dynamics in a 5,000-qubit programmable spin glass

Focus: D-Wave

D-Wave being the first company to file for a patent.

D-Wave gained a lot of notice once multi-qubit quantum tunneling effects were observed experimental and showed the computational potential it may have.

Their most advanced is the 5,760-qubit Advantage machine with which the following study was performed:



Article | Published: 19 April 2023

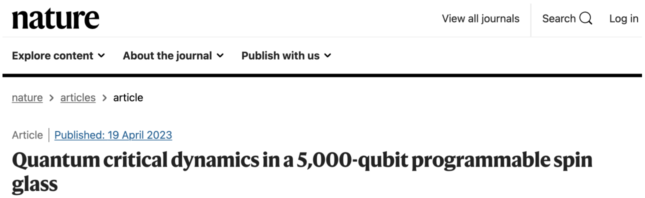
Quantum critical dynamics in a 5,000-qubit programmable spin glass

Focus: D-Wave

D-Wave being the first company to file for a patent.

D-Wave gained a lot of notice once multi-qubit quantum tunneling effects were observed experimental and showed the computational potential it may have.

Their most advanced is the 5,760-qubit Advantage machine with which the following study was performed:



nature

Article | Published: 19 April 2023

Quantum critical dynamics in a 5,000-qubit programmable spin glass

Measuring Performance

It is commong to use a metric known as Time-To-Solution (TTS) when performing benchmarking studies.

Data collected from multiple runs of the QA are used to compute the probability of finding a ground state solution for the given configuration of (adjustable) parameters. This probability is:

$$p_{\mathrm{TTS}} := \frac{\#\text{ of ground state solutions}}{\#\text{ of total QA runs}}. \tag{1.1}$$

Measuring Performance

It is commong to use a metric known as Time-To-Solution (TTS) when performing benchmarking studies.

Data collected from multiple runs of the QA are used to compute the probability of finding a ground state solution for the given configuration of (adjustable) parameters. This probability is:

$$p_{\mathrm{TTS}} := \frac{\#\text{ of ground state solutions}}{\#\text{ of total QA runs}}. \tag{1.1}$$

Measuring Performance

It is commong to use a metric known as Time-To-Solution (TTS) when performing benchmarking studies.

Data collected from multiple runs of the QA are used to compute the probability of finding a ground state solution for the given configuration of (adjustable) parameters. This probability is:

$$p_{\mathrm{TTS}} := \frac{\#\text{ of ground state solutions}}{\#\text{ of total QA runs}}. \tag{1.1}$$

TTS proper

The TTS proper is defined as the expected time to obtain the ground state solution at least once with success probability α and it is computed as:

$$\mathrm{TTS} = t_{\mathrm{run}} \frac{1 - \log \alpha}{1 - \log p_{\mathrm{TTS}}}. \tag{1.2}$$

Here t_{run} is the annealing time for a single run of the QA and $\alpha = 0.99$ by default.

Scheduling is a NP-Hard problem and you should expect that TTS scales exponentially with the size of the input N.

TTS proper

The TTS proper is defined as the expected time to obtain the ground state solution at least once with success probability α and it is computed as:

$$\mathrm{TTS} = t_{\mathrm{run}} \frac{1 - \log \alpha}{1 - \log p_{\mathrm{TTS}}}. \tag{1.2}$$

Here t_{run} is the annealing time for a single run of the QA and $\alpha = 0.99$ by default.

Scheduling is a NP-Hard problem and you should expect that TTS scales exponentially with the size of the input N.

TTS proper

The TTS proper is defined as the expected time to obtain the ground state solution at least once with success probability α and it is computed as:

$$\mathrm{TTS} = t_{\mathrm{run}} \frac{1 - \log \alpha}{1 - \log p_{\mathrm{TTS}}}. \tag{1.2}$$

Here t_{run} is the annealing time for a single run of the QA and $\alpha = 0.99$ by default.

Scheduling is a NP-Hard problem and you should expect that TTS scales exponentially with the size of the input N.

TTS proper

The TTS proper is defined as the expected time to obtain the ground state solution at least once with success probability α and it is computed as:

$$\mathrm{TTS} = t_{\mathrm{run}} \frac{1 - \log \alpha}{1 - \log p_{\mathrm{TTS}}}. \tag{1.2}$$

Here t_{run} is the annealing time for a single run of the QA and $\alpha = 0.99$ by default.

Scheduling is a NP-Hard problem and you should expect that TTS scales exponentially with the size of the input N.

TTS proper

Note that, in practice, when measuring TTS several aspects need to be considered (many of which are true for other models of quantum computation too), mainly:

- problem preparation time
- annealing time
- readout time
- repetition

TTS proper

Note that, in practice, when measuring TTS several aspects need to be considered (many of which are true for other models of quantum computation too), mainly:

- problem preparation time
- annealing time
- readout time
- repetition

Exponential scaling

Quantum optimizers such as QA (are hoped to) solve NP-Hard combinatorial optimization problems in time proportional to $\exp(\beta N\gamma)$ as $N \to \infty$, for positive coefficients β (scaling exponent) and γ.

Reasonable to expect (since scheduling problems are NP-Hard) TTS should scale exponentially with the problem size N in the asymptotic limit for $\gamma = 1$.

The value of the β parameter that turns out to fit the experimental results $\mathrm{TTS} = T_0 \exp \beta N$, for some constant $T_0 > 0$, ranges between 1.01 and 1.17 depending on the D-Wave machine.

Exponential scaling

Quantum optimizers such as QA (are hoped to) solve NP-Hard combinatorial optimization problems in time proportional to exp($\beta N \gamma$) as $N \to \infty$, for positive coefficients β (scaling exponent) and γ.

Reasonable to expect (since scheduling problems are NP-Hard) TTS should scale exponentially with the problem size N in the asymptotic limit for $\gamma = 1$.

The value of the β parameter that turns out to fit the experimental results $\mathrm{TTS} = T_0 \exp \beta N$, for some constant $T_0 > 0$, ranges between 1.01 and 1.17 depending on the D-Wave machine.

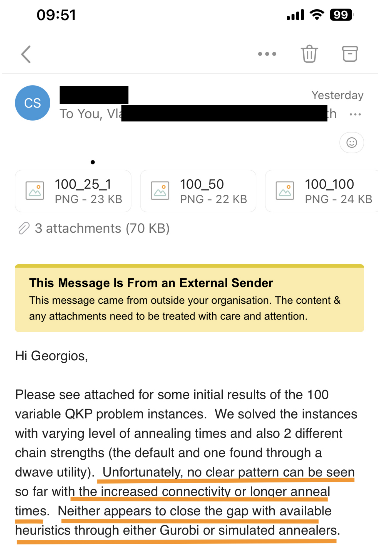
Exponential scaling

Quantum optimizers such as QA (are hoped to) solve NP-Hard combinatorial optimization problems in time proportional to $\exp(\beta N\gamma)$ as $N \to \infty$, for positive coefficients β (scaling exponent) and γ.

Reasonable to expect (since scheduling problems are NP-Hard) TTS should scale exponentially with the problem size N in the asymptotic limit for $\gamma = 1$.

The value of the β parameter that turns out to fit the experimental results $\mathrm{TTS} = T_0 \exp \beta N$, for some constant $T_0 > 0$, ranges between 1.01 and 1.17 depending on the D-Wave machine.

It's a tough job



09:51

[redacted]

Yesterday

To You, Vl[redacted]h

100_25_1
PNG - 23 KB

100_50
PNG - 22 KB

100_100
PNG - 24 KB

3 attachments (70 KB)

This Message Is From an External Sender
This message came from outside your organisation. The content & any attachments need to be treated with care and attention.

Hi Georgios,

Please see attached for some initial results of the 100 variable QKP problem instances. We solved the instances with varying level of annealing times and also 2 different chain strengths (the default and one found through a dwave utility). Unfortunately, no clear pattern can be seen so far with the increased connectivity or longer anneal times. Neither appears to close the gap with available heuristics through either Gurobi or simulated annealers.

What can we hope from QA?

The conclusion is that the experimental results align with the expectation that QA (and similar quantum optimization techniques) **may** be able to solve NP-Hard combinatorial optimization problems with an exponential time complexity.

The fact that β is close to 1 might indicate a near-linear scaling with problem size, which is promising for solving large optimization problems.

However, even with this seemingly near-linear scaling, the time and resources required can still be large for sufficiently big problem sizes.

What can we hope from QA?

The conclusion is that the experimental results align with the expectation that QA (and similar quantum optimization techniques) **may** be able to solve NP-Hard combinatorial optimization problems with an exponential time complexity.

The fact that β is close to 1 might indicate a near-linear scaling with problem size, which is promising for solving large optimization problems.

However, even with this seemingly near-linear scaling, the time and resources required can still be large for sufficiently big problem sizes.

What can we hope from QA?

The conclusion is that the experimental results align with the expectation that QA (and similar quantum optimization techniques) **may** be able to solve NP-Hard combinatorial optimization problems with an exponential time complexity.

The fact that β is close to 1 might indicate a near-linear scaling with problem size, which is promising for solving large optimization problems.

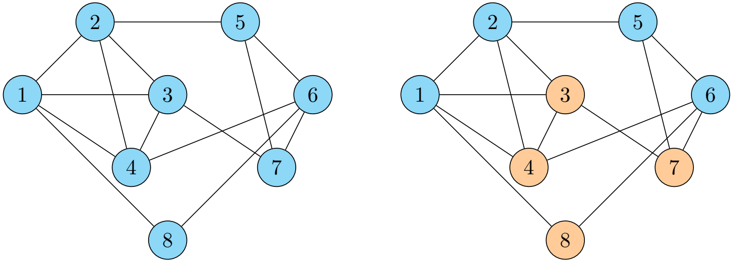
However, even with this seemingly near-linear scaling, the time and resources required can still be large for sufficiently big problem sizes.

More on QUBO

In this section we aim to discuss a few more QUBO formulations of interesting hard problems.

Graph Partitioning

Consider an undirected graph $G = (V, E)$. The task is to partition the set of vertices V into two subsets of equal size $N/2$, such that the number of edges connecting the two subsets is minimized.



Graph Partitioning

We can directly assign spin variables represented by the graph vertices where $x = +1$ values mean the blue class and $x = -1$ values mean the orange class.

The problem is solved by considering the following cost function:

$$L(x) = L_A(x) + L_B(x) \tag{2.1}$$

Graph Partitioning

We can directly assign spin variables represented by the graph vertices where $x = +1$ values mean the blue class and $x = -1$ values mean the orange class.

The problem is solved by considering the following cost function:

$$L(x) = L_A(x) + L_B(x) \tag{2.1}$$

Graph Partitioning

$$L_A(x) = \alpha \sum_{i=1}^{N} x_i, \tag{2.2}$$

This term provides a penalty term if the number of elements in the blue set is not equal to the number of elements in the orange set.

Graph Partitioning

$$L_B(x) = \beta \sum_{(u,v)\in E(G)} \frac{1 - x_u x_v}{2}, \tag{2.3}$$

This term provides a penalty each time an edge connects vertices from different subsets.

- $\beta > 0$: wish to minimize the number of edges between the two subsets
- $\beta < 0$: must be small enough so that it is never favorable to violate the other constraint L_A

Graph Partitioning

$$L_B(x) = \beta \sum_{(u,v)\in E(G)} \frac{1 - x_u x_v}{2}, \tag{2.3}$$

This term provides a penalty each time an edge connects vertices from different subsets.

- $\beta > 0$: wish to minimize the number of edges between the two subsets
- $\beta < 0$: must be small enough so that it is never favorable to violate the other constraint L_A

Binary Integer Linear Programming

Consider the binary vector $x = (x_1 \dots x_N) \in \{0,1\}^N$. Binary integer linear programming (BILP) amounts to the following problem:

$$\begin{aligned} \max_{x \in \{0,1\}^N} \quad & cx \\ \text{s.t.} \quad & Ax = b \\ & A \in \mathbb{R}^{M \times N} \\ & b \in \mathbb{R}^M \end{aligned} \tag{2.4}$$

Binary Integer Linear Programming

Consider the binary vector $x = (x_1 \dots x_N) \in \{0,1\}^N$. Binary integer linear programming (BILP) amounts to the following problem:

$$\begin{aligned} \max_{x \in \{0,1\}^N} \quad & cx \\ \text{s.t.} \quad & Ax = b \\ & A \in \mathbb{R}^{M \times N} \\ & b \in \mathbb{R}^M \end{aligned} \tag{2.4}$$

Binary Integer Linear Programming

A variety of problems can be formed as BILPs (for example in the context of banking revenue maximization subject to regulating constraints).

The cost function $L(x)$ corresponding to the QUBO formulation is

$$L(x) = L_A(x) + L_B(x), \tag{2.5}$$

Binary Integer Linear Programming

A variety of problems can be formed as BILPs (for example in the context of banking revenue maximization subject to regulating constraints).

The cost function $L(x)$ corresponding to the QUBO formulation is

$$L(x) = L_A(x) + L_B(x), \tag{2.5}$$

Binary Integer Linear Programming

$$L_A(x) = \alpha \sum_{j=1}^{m} \left(b_j - \sum_{i=1}^{N} A_{ij} x_i \right)^2 , \tag{2.6}$$

α is a constant.

Note that $L_A(x) = 0$ enforces the constraint $Ax = b$. When this is not met, we get an overall penalty to the objective function.

Binary Integer Linear Programming

$$L_A(x) = \alpha \sum_{j=1}^{m} \left(b_j - \sum_{i=1}^{N} A_{ij} x_i \right)^2 , \tag{2.6}$$

α is a constant.

Note that $L_A(x) = 0$ enforces the constraint $Ax = b$. When this is not met, we get an overall penalty to the objective function.

Binary Integer Linear Programming

$$L_B(x) = -\beta \sum_{i=1}^{N} c_i x_i, \tag{2.7}$$

for $\beta < \alpha$ another constant.

onstants α and β are tuning parameter that determines the relative importance of maximizing the objective function compared to satisfying the constraints.

The condition $\beta < \alpha$ ensures that the constraints take precedence over the objective function, which is usually the case in constrained optimization problems.

The reason for the minus sign is?

Binary Integer Linear Programming

$$L_B(x) = -\beta \sum_{i=1}^{N} c_i x_i, \tag{2.7}$$

for $\beta < \alpha$ another constant.

onstants α and β are tuning parameter that determines the relative importance of maximizing the objective function compared to satisfying the constraints.

The condition $\beta < \alpha$ ensures that the constraints take precedence over the objective function, which is usually the case in constrained optimization problems.

The reason for the minus sign is?

Binary Integer Linear Programming

$$L_B(x) = -\beta \sum_{i=1}^{N} c_i x_i, \tag{2.7}$$

for $\beta < \alpha$ another constant.

onstants α and β are tuning parameter that determines the relative importance of maximizing the objective function compared to satisfying the constraints.

The condition $\beta < \alpha$ ensures that the constraints take precedence over the objective function, which is usually the case in constrained optimization problems.

The reason for the minus sign is?

Binary Integer Linear Programming

$$L_B(x) = -\beta \sum_{i=1}^{N} c_i x_i, \tag{2.7}$$

for $\beta < \alpha$ another constant.

onstants α and β are tuning parameter that determines the relative importance of maximizing the objective function compared to satisfying the constraints.

The condition $\beta < \alpha$ ensures that the constraints take precedence over the objective function, which is usually the case in constrained optimization problems.

The reason for the minus sign is?

Portfolio optimization

One of the fundamental problems in quantitative finance is portfolio optimization which is part of modern portfolio theory (MPT).

A typical portfolio optimization is formulated as follows:

- N number of assets (things you can buy or sell in a market)
- μ_i expected return of asset $i \in [N]$
- σ_{ij} the covariance between the returns of asset i and asset j
- R the target portfolio return

Portfolio optimization

One of the fundamental problems in quantitative finance is portfolio optimization which is part of modern portfolio theory (MPT).

A typical portfolio optimization is formulated as follows:

- N number of assets (things you can buy or sell in a market)
- μ_i expected return of asset $i \in [N]$
- σ_{ij} the covariance between the returns of asset i and asset j
- R the target portfolio return

Portfolio optimization

One of the fundamental problems in quantitative finance is portfolio optimization which is part of modern portfolio theory (MPT).

A typical portfolio optimization is formulated as follows:

- N number of assets (things you can buy or sell in a market)
- μ_i expected return of asset $i \in [N]$
- σ_{ij} the covariance between the returns of asset i and asset j
- R the target portfolio return

Portfolio optimization

The decision variables are the weights $w \in \mathbb{R}^N$.

The standard approach here is the Markowitz mean-variance approach. This amounts to the following quadratic program:

$$\begin{aligned} \min_{w \in \mathbb{R}^N} \quad & \sum_{i,j=1}^{N} w_i w_j \sigma_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{N} w_i = 1, \\ & \sum_{i=1}^{N} w_i \mu_i = R. \end{aligned} \tag{2.8}$$

Portfolio optimization

The decision variables are the weights $w \in \mathbb{R}^N$.

The standard approach here is the Markowitz mean-variance approach. This amounts to the following quadratic program:

$$\begin{aligned} \min_{w \in \mathbb{R}^N} \quad & \sum_{i,j=1}^{N} w_i w_j \sigma_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{N} w_i = 1, \\ & \sum_{i=1}^{N} w_i \mu_i = R. \end{aligned} \tag{2.8}$$

Portfolio optimization

The decision variables are the weights $w \in \mathbb{R}^N$.

The standard approach here is the Markowitz mean-variance approach. This amounts to the following quadratic program:

$$\begin{aligned} \min_{w \in \mathbb{R}^N} \quad & \sum_{i,j=1}^{N} w_i w_j \sigma_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{N} w_i = 1, \\ & \sum_{i=1}^{N} w_i \mu_i = R. \end{aligned} \tag{2.8}$$

Portfolio optimization
Intuition

Problem amounts: construction of an optimal portfolio from the set of all possible assets with known characteristics such as their returns, volatilities, and pairwise correlations.

Expect to select $M \leq N$ assets from the set of available N assets that should be the best possible choice according to the criteria set by the constraints.

Portfolio optimization
Intuition

Problem amounts: construction of an optimal portfolio from the set of all possible assets with known characteristics such as their returns, volatilities, and pairwise correlations.

Expect to select $M \leq N$ assets from the set of available N assets that should be the best possible choice according to the criteria set by the constraints.

Portfolio optimization

Consider the case where where weights w are discrete; this situation starts resembling like a NP-Complete problem.

The previous problem can be mapped to a QUBO suitable for QA. This is done as follows.

We define the QUBO objective as:

$$L(x) = \sum_{i=1}^{N} a_i x_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} b_{ij} x_i x_j. \tag{2.9}$$

Portfolio optimization

Consider the case where where weights w are discrete; this situation starts resembling like a NP-Complete problem.

The previous problem can be mapped to a QUBO suitable for QA. This is done as follows.

We define the QUBO objective as:

$$L(x) = \sum_{i=1}^{N} a_i x_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} b_{ij} x_i x_j. \tag{2.9}$$

Portfolio optimization

Consider the case where where weights w are discrete; this situation starts resembling like a NP-Complete problem.

The previous problem can be mapped to a QUBO suitable for QA. This is done as follows.

We define the QUBO objective as:

$$L(x) = \sum_{i=1}^{N} a_i x_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} b_{ij} x_i x_j. \tag{2.9}$$

Portfolio optimization

Consider the case where where weights w are discrete; this situation starts resembling like a NP-Complete problem.

The previous problem can be mapped to a QUBO suitable for QA. This is done as follows.

We define the QUBO objective as:

$$L(x) = \sum_{i=1}^{N} a_i x_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} b_{ij} x_i x_j. \tag{2.9}$$

Portfolio optimization

In this context

$$x_i = \begin{cases} 1 & \text{means asset } i \text{ is selected,} \\ 0 & \text{means asset } i \text{ is not selected.} \end{cases} \tag{2.10}$$

Then, given the N asset set $x = \{x_1, \dots, x_N\}$ find the binary configuration that minimizes the $L(x)$ subject to the cardinality constraint that can be added via a penalty term $L_{\mathrm{pen}}(x)$.

Portfolio optimization

In this context

$$x_i = \begin{cases} 1 & \text{means asset } i \text{ is selected,} \\ 0 & \text{means asset } i \text{ is not selected.} \end{cases} \tag{2.10}$$

Then, given the N asset set $x = \{x_1, \ldots, x_N\}$ find the binary configuration that minimizes the $L(x)$ subject to the cardinality constraint that can be added via a penalty term $L_{\mathrm{pen}}(x)$.

Portfolio optimization

Specifically, the requirement that $\sum_{i=1}^{N} x_i = M$ is encoded via:

$$L_{\text{pen}}(x) = P\left(M - \sum_{i=1}^{N} x_i\right)^2 \tag{2.11}$$

Coefficients a_i reflect the asset attractiveness as a standalone (think user defined hyperparameter).

Assets with large expected risk-adjusted returns rewarded with negative values for a_i; assets with small expected risk-adjusted returns penalised with positive values of a_i.

b_{ij}: pairwise diversification penalties (if positive) and rewards (if negative) For all purposes of this course, assume a_i and b_{ij} as given.

Portfolio optimization

Specifically, the requirement that $\sum_{i=1}^{N} x_i = M$ is encoded via:

$$L_{\text{pen}}(x) = P\left(M - \sum_{i=1}^{N} x_i\right)^2 \tag{2.11}$$

Coefficients a_i reflect the asset attractiveness as a standalone (think user defined hyperparameter).

Assets with large expected risk-adjusted returns rewarded with negative values for a_i; assets with small expected risk-adjusted returns penalised with positive values of a_i.

b_{ij}: pairwise diversification penalties (if positive) and rewards (if negative) For all purposes of this course, assume a_i and b_{ij} as given.

Portfolio optimization

Specifically, the requirement that $\sum_{i=1}^{N} x_i = M$ is encoded via:

$$L_{\text{pen}}(x) = P\left(M - \sum_{i=1}^{N} x_i\right)^2 \tag{2.11}$$

Coefficients a_i reflect the asset attractiveness as a standalone (think user defined hyperparameter).

Assets with large expected risk-adjusted returns rewarded with negative values for a_i; assets with small expected risk-adjusted returns penalised with positive values of a_i.

b_{ij}: pairwise diversification penalties (if positive) and rewards (if negative) For all purposes of this course, assume a_i and b_{ij} as given.

Portfolio optimization

Specifically, the requirement that $\sum_{i=1}^{N} x_i = M$ is encoded via:

$$L_{\text{pen}}(x) = P\left(M - \sum_{i=1}^{N} x_i\right)^2 \tag{2.11}$$

Coefficients a_i reflect the asset attractiveness as a standalone (think user defined hyperparameter).

Assets with large expected risk-adjusted returns rewarded with negative values for a_i; assets with small expected risk-adjusted returns penalised with positive values of a_i.

b_{ij}: pairwise diversification penalties (if positive) and rewards (if negative) For all purposes of this course, assume a_i and b_{ij} as given.

Portfolio optimization

Specifically, the requirement that $\sum_{i=1}^{N} x_i = M$ is encoded via:

$$L_{\text{pen}}(x) = P\left(M - \sum_{i=1}^{N} x_i\right)^2 \tag{2.11}$$

Coefficients a_i reflect the asset attractiveness as a standalone (think user defined hyperparameter).

Assets with large expected risk-adjusted returns rewarded with negative values for a_i; assets with small expected risk-adjusted returns penalised with positive values of a_i.

b_{ij}: pairwise diversification penalties (if positive) and rewards (if negative) For all purposes of this course, assume a_i and b_{ij} as given.

Portfolio optimization

Specifically, the requirement that $\sum_{i=1}^{N} x_i = M$ is encoded via:

$$L_{\text{pen}}(x) = P\left(M - \sum_{i=1}^{N} x_i\right)^2 \tag{2.11}$$

Coefficients a_i reflect the asset attractiveness as a standalone (think user defined hyperparameter).

Assets with large expected risk-adjusted returns rewarded with negative values for a_i; assets with small expected risk-adjusted returns penalised with positive values of a_i.

b_{ij}: pairwise diversification penalties (if positive) and rewards (if negative) For all purposes of this course, assume a_i and b_{ij} as given.

Portfolio optimization

The total QUBO to be solved is:

$$\min_{x\in\{0,1\}^N} \quad L_{\mathrm{total}}(x) := L(x) + L_{\mathrm{pen}}(x) \tag{2.12}$$

Portfolio optimization

Intuition

The minimization of this QUBO optimizes for the risk-adjusted returns by using the so-called Sharpe ratio: The ratio describes how much excess return you receive for the extra volatility you endure for holding a riskier asset.

This is computed as $(r - r_0)/\sigma$ where r is the expected (annualised) asset return, r_0 is the applicable risk-free interest rate and σ is the asset volatility.

Expected returns can be either estimated as the historical returns or derived independently using e.g. Monte Carlo simulations.

Portfolio optimization
Intuition

The minimization of this QUBO optimizes for the risk-adjusted returns by using the so-called Sharpe ratio: The ratio describes how much excess return you receive for the extra volatility you endure for holding a riskier asset.

This is computed as $(r - r_0)/\sigma$ where r is the expected (annualised) asset return, r_0 is the applicable risk-free interest rate and σ is the asset volatility.

Expected returns can be either estimated as the historical returns or derived independently using e.g. Monte Carlo simulations.

Portfolio optimization

Intuition

The minimization of this QUBO optimizes for the risk-adjusted returns by using the so-called Sharpe ratio: The ratio describes how much excess return you receive for the extra volatility you endure for holding a riskier asset.

This is computed as $(r - r_0)/\sigma$ where r is the expected (annualised) asset return, r_0 is the applicable risk-free interest rate and σ is the asset volatility.

Expected returns can be either estimated as the historical returns or derived independently using e.g. Monte Carlo simulations.

QBoost

Let us discuss how QBoost is used in the context of Machine Learning (ML).
First, let us set up some notation:

Object	Definition
$x_t \in \mathbb{R}^N$	vector of N features
$y_t \in \{0, 1\}$	binary classification label
$\{x_t, y_t\}_{t \in [M]}$	training set
$c_i(x_t) = \pm\frac{1}{N}$	value of weak classifier i on event t
$q := (q_1, \ldots, q_N)$	vector of binary weights associated with each weak classifier

QBoost

Let us discuss how QBoost is used in the context of Machine Learning (ML). First, let us set up some notation:

Object	Definition
$x_t \in \mathbb{R}^N$	vector of N features
$y_t \in \{0, 1\}$	binary classification label
$\{x_t, y_t\}_{t\in[M]}$	training set
$c_i(x_t) = \pm\frac{1}{N}$	value of weak classifier i on event t
$q := (q_1, \dots, q_N)$	vector of binary weights associated with each weak classifier

QBoost

The classification error for sample t is given by the square error

$$\left(\sum_{i=1}^{N} c_i(x_t)q_i - y_t\right)^2 . \tag{3.1}$$

QBoost

The classification error for sample t is given by the square error

$$\left(\sum_{i=1}^{N} c_i(x_t) q_i - y_t \right)^2 . \tag{3.1}$$

QBoost

The total cost function to minimize is the sum of squared errors across the training data:

$$L(s) = \sum_{t=1}^{M} \left(\sum_{i=1}^{N} c_i(x_t) s_i - y_t \right)^2 \tag{3.2}$$

Expanding yields a term y_t^2 that does not depend on s and does not influence the minimization of L (can be absorbed as a constant energy shift).

QBoost

The total cost function to minimize is the sum of squared errors across the training data:

$$L(s) = \sum_{t=1}^{M} \left(\sum_{i=1}^{N} c_i(x_t) s_i - y_t \right)^2 \tag{3.2}$$

Expanding yields a term y_t^2 that does not depend on s and does not influence the minimization of L (can be absorbed as a constant energy shift).

QBoost

Overfitting can be done by adding a penalty $\lambda > 0$. The objective to minimize in the QBoost agorithm is:

$$\tilde{L}(s) = \sum_{t=1}^{M} \left(\sum_{i=1}^{N} c_i(x_t) q_i \sum_{j=1}^{N} c_j(x_t) s_j - 2y_t \sum_{i=1}^{N} c_i(x_t) s_i \right) + \lambda \sum_{i=1}^{N} s_i \tag{3.3}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij} q_i q_j + \sum_{i=1}^{N} (\lambda - 2C_i) s_i, \tag{3.4}$$

QBoost

Overfitting can be done by adding a penalty $\lambda > 0$. The objective to minimize in the QBoost agorithm is:

$$\tilde{L}(s) = \sum_{t=1}^{M} \left(\sum_{i=1}^{N} c_i(x_t) q_i \sum_{j=1}^{N} c_j(x_t) s_j - 2y_t \sum_{i=1}^{N} c_i(x_t) s_i \right) + \lambda \sum_{i=1}^{N} s_i \tag{3.3}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij} q_i q_j + \sum_{i=1}^{N} (\lambda - 2C_i) s_i, \tag{3.4}$$

QBoost

$$C_{ij} := \sum_{t=1}^{M} c_i(x_t) c_j(x_t), \qquad C_i := \sum_{i=1}^{M} c_i(x_t) y_t.$$

Remark: the penalty term added here is analogous to LASSO regression method with L_1 penalty. Note that ridge regression with L_2 penalty could be chosen instead.

QBoost

$$C_{ij} := \sum_{t=1}^{M} c_i(x_t)c_j(x_t), \qquad C_i := \sum_{i=1}^{M} c_i(x_t)y_t.$$

Remark: the penalty term added here is analogous to LASSO regression method with L_1 penalty. Note that ridge regression with L_2 penalty could be chosen instead.

QBoost

Next, we need to map the problem to an Ising model. Consider σ to be spin variables by defining

$$\sigma = 2s - 1 \tag{3.5}$$

The Ising Hamiltonian is then written as:

$$H = \frac{1}{4}\sum_{i,j=1}^{N} C_{ij}\sigma_i\sigma_j + \frac{1}{2}\sum_{i,j=1}^{N} C_i\sigma_i + \sum_{i=1}^{N}(\lambda' - C_i)\sigma_i, \tag{3.6}$$

($\lambda' := \frac{1}{2}\lambda$ is a rescaled penalty coefficient)

QBoost

Next, we need to map the problem to an Ising model. Consider σ to be spin variables by defining

$$\sigma = 2s - 1 \tag{3.5}$$

The Ising Hamiltonian is then written as:

$$H = \frac{1}{4}\sum_{i,j=1}^{N} C_{ij}\sigma_i\sigma_j + \frac{1}{2}\sum_{i,j=1}^{N} C_i\sigma_i + \sum_{i=1}^{N}(\lambda' - C_i)\sigma_i, \tag{3.6}$$

($\lambda' := \frac{1}{2}\lambda$ is a rescaled penalty coefficient)

QBoost

Next, we need to map the problem to an Ising model. Consider σ to be spin variables by defining

$$\sigma = 2s - 1 \tag{3.5}$$

The Ising Hamiltonian is then written as:

$$H = \frac{1}{4}\sum_{i,j=1}^{N} C_{ij}\sigma_i\sigma_j + \frac{1}{2}\sum_{i,j=1}^{N} C_i\sigma_i + \sum_{i=1}^{N}(\lambda' - C_i)\sigma_i, \tag{3.6}$$

($\lambda' := \frac{1}{2}\lambda$ is a rescaled penalty coefficient)

QBoost

Next, we need to map the problem to an Ising model. Consider σ to be spin variables by defining

$$\sigma = 2s - 1 \tag{3.5}$$

The Ising Hamiltonian is then written as:

$$H = \frac{1}{4}\sum_{i,j=1}^{N} C_{ij}\sigma_i\sigma_j + \frac{1}{2}\sum_{i,j=1}^{N} C_i\sigma_i + \sum_{i=1}^{N}(\lambda' - C_i)\sigma_i, \tag{3.6}$$

($\lambda' := \frac{1}{2}\lambda$ is a rescaled penalty coefficient)

QBoost

QA aims to solve the problem to minimize H and compute the ground state spin configuration bit-string $|s\rangle$, with $s \in \{-1, 1\}^N$.

For each new sample x, the classifier is given as

$$R(x) = \sum_{i=1}^{N} s_i c_i(x) \quad \in [-1, 1]. \tag{3.7}$$

QBoost

QA aims to solve the problem to minimize H and compute the ground state spin configuration bit-string $|s\rangle$, with $s \in \{-1, 1\}^N$.

For each new sample x, the classifier is given as

$$R(x) = \sum_{i=1}^{N} s_i c_i(x) \quad \in [-1, 1]. \tag{3.7}$$

QBoost: Application

QA for ML applications has been gaining a lot of popularity (and serves as a business model for a number of quantum computing startups).

It is claimed to have demonstrated performance advantage in compaerison with algorithms such as binary decision tree-based Extreme Gradient Boosting (XGBoost) and DNN classifiers on small datasets.

QBoost: Application

QA for ML applications has been gaining a lot of popularity (and serves as a business model for a number of quantum computing startups).

It is claimed to have demonstrated performance advantage in compaerison with algorithms such as binary decision tree-based Extreme Gradient Boosting (XGBoost) and DNN classifiers on small datasets.

QBoost: Application

A very interesting application is that of forecasting credit card client defaults. For that one can utilize a publicly available dataset available from the UCI Machine Learning Repository. This dataset consists of 30,000 samples with binary classifications:

- a client does not default - class 0
- a client does default - class 1

QBoost: Application

$N = 23$ features $(F_1, \dots, F_{23})$ available:

- F_1: amount of given credit (continuous)
- F_2: gender (binary)
- F_3: education (discrete)
- F_4: marital status (discrete)
- F_5: age (discrete)
- F_6: repayment status of previous month (discrete)
- F_7: repayment status of two months ago (discrete)
- F_8-F_{11}: similar (discrete)
- F_{12}: bill amount past month (continuous)
- F_{13}: bill amount two months ago (continuous)
- F_{14}-F_{17}: similar (continuous)
- F_{18}: amount of previous month payment (continuous)
- F_{19}: amount of payment two months ago (continuous)
- F_{20}-F_{23}: similar (continuous)

QBoost: Application Results

	Accuracy	Precision	Recall
GradBoost	0.83	0.69	0.35
MLP	0.83	0.69	0.35
QBoost	0.83	0.71	0.33

It has be argued that QBoost provides an improvement on such approaches by finding an optimal configuration of the weak classifiers.

QBoost: Application Results

	Accuracy	Precision	Recall
GradBoost	0.83	0.69	0.35
MLP	0.83	0.69	0.35
QBoost	0.83	0.71	0.33

It has be argued that QBoost provides an improvement on such approaches by finding an optimal configuration of the weak classifiers.

Break

Questions?

WS-QAOA

Quantum Approximate Optimization Algorithm (QAOA) encodes a combinatorial optimization problem in a Hamiltonian H_C whose ground state is the optimum solution.

QAOA first creates an initial state which is the ground state of a mixer Hamiltonian H_M where a common choice is

$$H_M = -\sum_{i=1}^{N} \sigma_i^x, \tag{4.1}$$

with ground state being $|+\rangle^{\otimes n}$.

WS-QAOA

Quantum Approximate Optimization Algorithm (QAOA) encodes a combinatorial optimization problem in a Hamiltonian H_C whose ground state is the optimum solution.

QAOA first creates an initial state which is the ground state of a mixer Hamiltonian H_M where a common choice is

$$H_M = -\sum_{i=1}^{N} \sigma_i^x, \tag{4.1}$$

with ground state being $|+\rangle^{\otimes n}$.

WS-QAOA

Then, recall, for depth-L QAOA, we apply L times the unitary $U_{\mathrm{QAOA}} = U_C(\gamma)U_B(\beta)$ defined as:

$$U_C(\gamma) := e^{-\imath\gamma H_C} \tag{4.2}$$

$$U_B(\beta) := e^{-\imath\beta H_M}. \tag{4.3}$$

The result is:

$$U_{\mathrm{QAOA}} \left|+\right\rangle^{\otimes n} = \left|\gamma, \beta\right\rangle . \tag{4.4}$$

A classical optimizer (e.g. SPSA) then seeks the optimal values of β and γ to create a trial state which minimizes the energy of the problem Hamiltonian H_C.

WS-QAOA

Then, recall, for depth-L QAOA, we apply L times the unitary $U_{\mathrm{QAOA}} = U_C(\gamma)U_B(\beta)$ defined as:

$$U_C(\gamma) := e^{-\imath\gamma H_C} \tag{4.2}$$

$$U_B(\beta) := e^{-\imath\beta H_M}. \tag{4.3}$$

The result is:

$$U_{\mathrm{QAOA}} \left|+\right\rangle^{\otimes n} = \left|\gamma, \beta\right\rangle . \tag{4.4}$$

A classical optimizer (e.g. SPSA) then seeks the optimal values of β and γ to create a trial state which minimizes the energy of the problem Hamiltonian H_C.

WS-QAOA

Then, recall, for depth-L QAOA, we apply L times the unitary $U_{\mathrm{QAOA}} = U_C(\gamma)U_B(\beta)$ defined as:

$$U_C(\gamma) := e^{-\imath\gamma H_C} \tag{4.2}$$

$$U_B(\beta) := e^{-\imath\beta H_M}. \tag{4.3}$$

The result is:

$$U_{\mathrm{QAOA}} \left|+\right\rangle^{\otimes n} = \left|\gamma, \beta\right\rangle . \tag{4.4}$$

A classical optimizer (e.g. SPSA) then seeks the optimal values of β and γ to create a trial state which minimizes the energy of the problem Hamiltonian H_C.

WS-QAOA

Then, recall, for depth-L QAOA, we apply L times the unitary $U_{\mathrm{QAOA}} = U_C(\gamma)U_B(\beta)$ defined as:

$$U_C(\gamma) := e^{-\imath\gamma H_C} \tag{4.2}$$

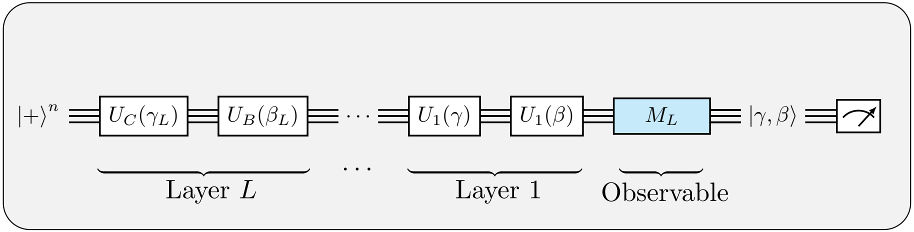
$$U_B(\beta) := e^{-\imath\beta H_M}. \tag{4.3}$$

The result is:

$$U_{\mathrm{QAOA}} \left|+\right\rangle^{\otimes n} = \left|\gamma, \beta\right\rangle. \tag{4.4}$$

A classical optimizer (e.g. SPSA) then seeks the optimal values of β and γ to create a trial state which minimizes the energy of the problem Hamiltonian H_C.

WS-QAOA



(Typo: instead of $U_C(\gamma_L)U_B(\beta_L)$ it should read instead $U_L(\gamma_L)U_L(\beta_L)$)

WS-QAOA

While very promising algorithm,

initially it lacked theoretical guarantees on its performance ratio and for certain problem instances of interest (e.g. Max-Cut) it cannot, for constant L, outperform the classical Goemans-Williamson randomized rounding approximation.

While several improvements of the QAOA have been developed in the literature, we will focus here on warm-starting QAOA of Egger et. al. (incl. Jakub).

WS-QAOA

For Max-Cut it finds cuts whose expected value is an α fraction of the global optimum, for $0.87856 < \alpha < 0.87857$, with the expectation over the randomization in the rounding procedure)

WS-QAOA

While very promising algorithm,

initially it lacked theoretical guarantees on its performance ratio and for certain problem instances of interest.

While several improvements of the QAOA have been developed in the literature, we will focus here on warm-starting QAOA of Egger et. al. (incl. Jakub).

WS-QAOA

While very promising algorithm,

initially it lacked theoretical guarantees on its performance ratio and for certain problem instances of interest.

While several improvements of the QAOA have been developed in the literature, we will focus here on warm-starting QAOA of Egger et. al. (incl. Jakub).

WS-QAOA

While very promising algorithm,

initially it lacked theoretical guarantees on its performance ratio and for certain problem instances of interest.

While several improvements of the QAOA have been developed in the literature, we will focus here on warm-starting QAOA of Egger et. al. (incl. Jakub).

WS-QAOA

Relaxations

QUBOs have already been discussed a lot. A common formulation is:

$$\min_{x \in \{0,1\}^n} x^T Q x + \mu^T x. \tag{4.5}$$

where x is a vector of n binary decision variables, $Q \in \mathbb{R}^{n \times n}$ a symmetric matrix, and $\mu \in \mathbb{R}^n$ a vector.

Since for binary variables $x_i^2 = x_i, \mu$ can be added to the diagonal of Σ, so add μ bcz it simplifies the notation.

Note that practically any mixed-integer linear program (MILP) can be encoded in a QUBO it is automatically NP-Hard.

WS-QAOA

Relaxations

QUBOs have already been discussed a lot. A common formulation is:

$$\min_{x \in \{0,1\}^n} x^T Q x + \mu^T x. \tag{4.5}$$

where x is a vector of n binary decision variables, $Q \in \mathbb{R}^{n \times n}$ a symmetric matrix, and $\mu \in \mathbb{R}^n$ a vector.

Since for binary variables $x_i^2 = x_i, \mu$ can be added to the diagonal of Σ, so add μ bcz it simplifies the notation.

Note that practically any mixed-integer linear program (MILP) can be encoded in a QUBO it is automatically NP-Hard.

WS-QAOA

Relaxations

QUBOs have already been discussed a lot. A common formulation is:

$$\min_{x \in \{0,1\}^n} x^T Q x + \mu^T x. \tag{4.5}$$

where x is a vector of n binary decision variables, $Q \in \mathbb{R}^{n \times n}$ a symmetric matrix, and $\mu \in \mathbb{R}^n$ a vector.

Since for binary variables $x_i^2 = x_i, \mu$ can be added to the diagonal of Σ, so add μ bcz it simplifies the notation.

Note that practically any mixed-integer linear program (MILP) can be encoded in a QUBO it is automatically NP-Hard.

WS-QAOA

If Q is positive semidefinite, there exists a trivial **continuous relaxation** of the QUBO above:

$$\min_{x\in[0,1]^n} x^T Qx \tag{4.6}$$

is a convex quadratic program and the optimal solution c^* of the continuous relaxation is easily obtainable with classical optimizers.

WS-QAOA

The solutions of continuous-valued relaxation discussed above can be used to initialize VQAs: this is known as warm-starting.

Let us focus on how to warm-start the QAOA of Fahri et. al.

WS-QAOA

The solutions of continuous-valued relaxation discussed above can be used to initialize VQAs: this is known as warm-starting.

Let us focus on how to warm-start the QAOA of Fahri et. al.

WS-QAOA

In QAOA, each decision variable x_i of the discrete optimization problem corresponds to a qubit by the substitution $x_i = (1 - s_i)/2$. Each s_i is replaced by a spin operator σ_i to transform the cost function to a cost Hamiltonian H_C.

After utilizing the unitary U_{QAOA}, one performs the final measurement: a randomized rounding. Warm-starting amounts to replacing the initial equal superposition state $|+\rangle^{\otimes n}$ with a state

$$|\phi^*\rangle = \bigotimes_{i=0}^{n-1} \hat{R}_y(\theta_i)\,|0\rangle_n \tag{4.7}$$

which corresponds to the solution c^* of the relaxed Problem (4.6).

WS-QAOA

In QAOA, each decision variable x_i of the discrete optimization problem corresponds to a qubit by the substitution $x_i = (1 - s_i)/2$. Each s_i is replaced by a spin operator σ_i to transform the cost function to a cost Hamiltonian H_C.

After utilizing the unitary U_{QAOA}, one performs the final measurement: a randomized rounding. Warm-starting amounts to replacing the initial equal superposition state $|+\rangle^{\otimes n}$ with a state

$$|\phi^*\rangle = \bigotimes_{i=0}^{n-1} \hat{R}_y(\theta_i)\,|0\rangle_n \qquad (4.7)$$

which corresponds to the solution c^* of the relaxed Problem (4.6).

WS-QAOA

In QAOA, each decision variable x_i of the discrete optimization problem corresponds to a qubit by the substitution $x_i = (1 - s_i)/2$. Each s_i is replaced by a spin operator σ_i to transform the cost function to a cost Hamiltonian H_C.

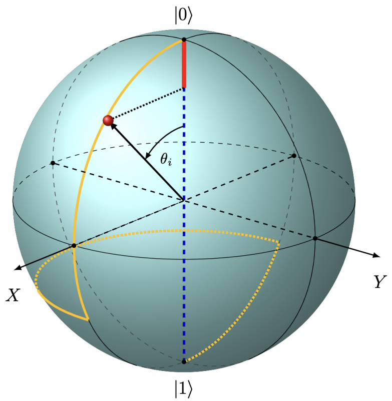
After utilizing the unitary U_{QAOA}, one performs the final measurement: a randomized rounding. Warm-starting amounts to replacing the initial equal superposition state $|+\rangle^{\otimes n}$ with a state

$$|\phi^*\rangle = \bigotimes_{i=0}^{n-1} \hat{R}_y(\theta_i)\,|0\rangle_n \tag{4.7}$$

which corresponds to the solution c^* of the relaxed Problem (4.6).

WS-QAOA

Here, $\hat{R}_y(\theta_i)$ is a θ_i-parametrized rotation around the y-axis of the qubit and $\theta_i := 2\arcsin(\sqrt{c_i^*})$ for c_i^* given as the solution of QUBO (4.6).



WS-QAOA

The mixer Hamiltonian also is replaced. A choice for the warm-starting mixer Hamiltonian is

$$H_M^{\mathrm{ws}} = \sum_{i=1}^{n} H_{M,i}^{\mathrm{ws}} \tag{4.8}$$

where

$$H_{M,i}^{\mathrm{ws}} = \begin{pmatrix} 2c_i^* - 1 & -2\sqrt{c_i^*(1-c_i^*)} \\ -2\sqrt{c_i^*(1-c_i^*)} & 1-2c_i^* \end{pmatrix} \tag{4.9}$$

which has $R_y(\theta_i)|0\rangle$ as ground state.One can show that the ground state of H_M^{ws} is $|\phi^*\rangle$ with energy $-n$. Therefore, WS-QAOA applies at layer k a mixing gate which is given by the time-evolved mixing Hamiltonian $U_M(\beta) = e^{-\imath\beta H_M^{\mathrm{ws}}}$.

WS-QAOA

The mixer Hamiltonian also is replaced. A choice for the warm-starting mixer Hamiltonian is

$$H_M^{\mathrm{ws}} = \sum_{i=1}^{n} H_{M,i}^{\mathrm{ws}} \tag{4.8}$$

where

$$H_{M,i}^{\mathrm{ws}} = \begin{pmatrix} 2c_i^* - 1 & -2\sqrt{c_i^*(1-c_i^*)} \\ -2\sqrt{c_i^*(1-c_i^*)} & 1 - 2c_i^* \end{pmatrix} \tag{4.9}$$

which has $R_y(\theta_i)\,|0\rangle$ as ground state.One can show that the ground state of H_M^{ws} is $|\phi^*\rangle$ with energy $-n$. Therefore, WS-QAOA applies at layer k a mixing gate which is given by the time-evolved mixing Hamiltonian $U_M(\beta) = e^{-\imath\beta H_M^{\mathrm{ws}}}$.

WS-QAOA

The mixer Hamiltonian also is replaced. A choice for the warm-starting mixer Hamiltonian is

$$H_M^{\text{ws}} = \sum_{i=1}^{n} H_{M,i}^{\text{ws}} \tag{4.8}$$

where

$$H_{M,i}^{\text{ws}} = \begin{pmatrix} 2c_i^* - 1 & -2\sqrt{c_i^*(1 - c_i^*)} \\ -2\sqrt{c_i^*(1 - c_i^*)} & 1 - 2c_i^* \end{pmatrix} \tag{4.9}$$

which has $R_y(\theta_i)\,|0\rangle$ as ground state. One can show that the ground state of H_M^{ws} is $|\phi^*\rangle$ with energy $-n$. Therefore, WS-QAOA applies at layer k a mixing gate which is given by the time-evolved mixing Hamiltonian $U_M(\beta) = e^{-\imath\beta H_M^{\text{ws}}}$.

WS-QAOA

The mixer Hamiltonian also is replaced. A choice for the warm-starting mixer Hamiltonian is

$$H_M^{\mathrm{ws}} = \sum_{i=1}^{n} H_{M,i}^{\mathrm{ws}} \tag{4.8}$$

where

$$H_{M,i}^{\mathrm{ws}} = \begin{pmatrix} 2c_i^* - 1 & -2\sqrt{c_i^*(1-c_i^*)} \\ -2\sqrt{c_i^*(1-c_i^*)} & 1 - 2c_i^* \end{pmatrix} \tag{4.9}$$

which has $R_y(\theta_i)\,|0\rangle$ as ground state.One can show that the ground state of H_M^{ws} is $|\phi^*\rangle$ with energy $-n$. Therefore, WS-QAOA applies at layer k a mixing gate which is given by the time-evolved mixing Hamiltonian $U_M(\beta) = e^{-\imath\beta H_M^{\mathrm{ws}}}$.

WS-QAOA

The mixer Hamiltonian also is replaced. A choice for the warm-starting mixer Hamiltonian is

$$H_M^{\mathrm{ws}} = \sum_{i=1}^{n} H_{M,i}^{\mathrm{ws}} \tag{4.8}$$

where

$$H_{M,i}^{\mathrm{ws}} = \begin{pmatrix} 2c_i^* - 1 & -2\sqrt{c_i^*(1-c_i^*)} \\ -2\sqrt{c_i^*(1-c_i^*)} & 1 - 2c_i^* \end{pmatrix} \tag{4.9}$$

which has $R_y(\theta_i)\left|0\right\rangle$ as ground state.One can show that the ground state of H_M^{ws} is $\left|\phi^*\right\rangle$ with energy $-n$. Therefore, WS-QAOA applies at layer k a mixing gate which is given by the time-evolved mixing Hamiltonian $U_M(\beta) = e^{-\imath\beta H_M^{\mathrm{ws}}}$.

WS-QAOA

For technical reasons one has to actually modify the definition of θ_i as

$$\begin{array}{lll} \theta_i = 2\arcsin\left(\sqrt{c_i^*}\right) & \text{if} & c_i^* \in [\varepsilon, 1-\varepsilon] \\ \theta_i = 2\arcsin(\sqrt{\varepsilon}) & \text{if} & c_i^* \leq \varepsilon \\ \theta_i = 2\arcsin(\sqrt{1-\varepsilon}) & \text{if} & c_i^* \geq 1-\varepsilon. \end{array}$$

where $\varepsilon \in [0, 0.5]$ and the mixer Hamiltonian H_M is adjusted accordingly.

The parameter ε provides a continuous mapping between WS-QAOA and standard QAOA since at $\varepsilon = 0.5$ the initial state is the equal superposition state and the mixer Hamiltonian is the X operator.

If all $c_i^* \in (0, 1)$ or $\varepsilon > 0$, WS-QAOA converges to the optimal solution of (QUBO) as the depth L approaches infinity as does standard QAOA.

WS-QAOA

For technical reasons one has to actually modify the definition of θ_i as

$$\begin{array}{lll} \theta_i = 2\arcsin\left(\sqrt{c_i^*}\right) & \text{if} & c_i^* \in [\varepsilon, 1-\varepsilon] \\ \theta_i = 2\arcsin(\sqrt{\varepsilon}) & \text{if} & c_i^* \leq \varepsilon \\ \theta_i = 2\arcsin(\sqrt{1-\varepsilon}) & \text{if} & c_i^* \geq 1-\varepsilon. \end{array}$$

where $\varepsilon \in [0, 0.5]$ and the mixer Hamiltonian H_M is adjusted accordingly.

The parameter ε provides a continuous mapping between WS-QAOA and standard QAOA since at $\varepsilon = 0.5$ the initial state is the equal superposition state and the mixer Hamiltonian is the X operator.

If all $c_i^* \in (0,1)$ or $\varepsilon > 0$, WS-QAOA converges to the optimal solution of (QUBO) as the depth L approaches infinity as does standard QAOA.

WS-QAOA

For technical reasons one has to actually modify the definition of θ_i as

$$\begin{array}{lll} \theta_i = 2\arcsin\left(\sqrt{c_i^*}\right) & \text{if} & c_i^* \in [\varepsilon, 1-\varepsilon] \\ \theta_i = 2\arcsin(\sqrt{\varepsilon}) & \text{if} & c_i^* \leq \varepsilon \\ \theta_i = 2\arcsin(\sqrt{1-\varepsilon}) & \text{if} & c_i^* \geq 1-\varepsilon. \end{array}$$

where $\varepsilon \in [0, 0.5]$ and the mixer Hamiltonian H_M is adjusted accordingly.

The parameter ε provides a continuous mapping between WS-QAOA and standard QAOA since at $\varepsilon = 0.5$ the initial state is the equal superposition state and the mixer Hamiltonian is the X operator.

If all $c_i^* \in (0, 1)$ or $\varepsilon > 0$, WS-QAOA converges to the optimal solution of (QUBO) as the depth L approaches infinity as does standard QAOA.

WS-QAOA

For large enough L, WS-QAOA the adiabatic evolution transforming the ground state of the mixer into the ground state of $\hat{H}_C$ as expected. The speed of the adiabatic evolution is limited by the spectral gap of the intermediate Hamiltonians as we discussed in the previous lecture.

The speed of the evolution can be related to the depth L, where a slow evolution (larger terminal time T) implies a larger L. The idea of WS-QAOA is to speed-up this evolution by optimizing the parameters instead of following a fixed annealing schedule.

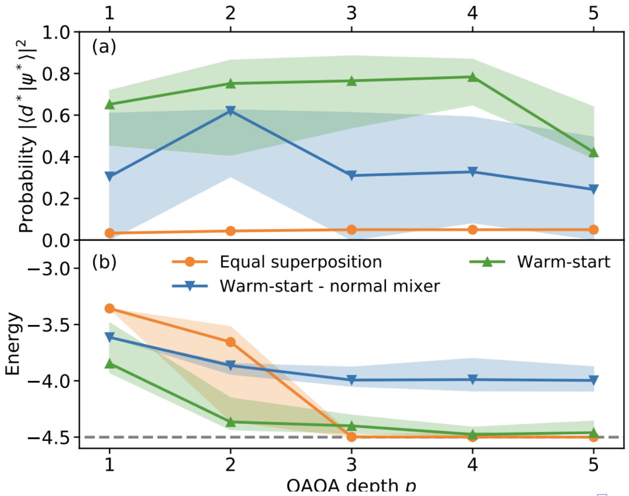
WS-QAOA

For large enough L, WS-QAOA the adiabatic evolution transforming the ground state of the mixer into the ground state of $\hat{H}_C$ as expected. The speed of the adiabatic evolution is limited by the spectral gap of the intermediate Hamiltonians as we discussed in the previous lecture.

The speed of the evolution can be related to the depth L, where a slow evolution (larger terminal time T) implies a larger L. The idea of WS-QAOA is to speed-up this evolution by optimizing the parameters instead of following a fixed annealing schedule.

WS-QAOA

Below we quote a nice experimental demonstration from Egger et. al.:



(a)
(b)
Probability $|\langle d^* | \psi^* \rangle|^2$
Energy
Equal superposition
Warm-start
Warm-start - normal mixer
QAOA depth p
1.0
0.8
0.6
0.4
0.2
0.0
−3.0
−3.5
−4.0
−4.5
1
2
3
4
5