CHAPTER 9

Applications in Financial Services

In this chapter, we want to make a few remarks on the practical aspects of use of what we have seen, esp. in Chapters 7 and 8, in the financial services industry. Therein, one needs to deal with many more vendors (i.e., salesmen) than universities (i.e., researchers).

1. Practical aspects of quantum annealers

Outside of many reputable vendors of gate-based quantum computers, there are also numerous vendors of so-called quantum annealers. While there are several quantum annealers across the world in academic environments, the most well-known vendor is D-Wave Systems. Other vendors that develop superconducting quantum annealers are Qilimanjaro and Avaqus. 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. Finally, several companies manufacture specialized classical hardware (e.g., based on FPGAs) that simulates quantum annealing, for example Fujitsu.

What is common among the above is that the most obvious use cases and candidate problems to be attacked by these machines are hard optimization problems. Therefore, similar to companies offering classical solvers, such as Gurobi, understanding of optimization is crucial for a career in this area.

Note that the applications go beyond optimization, e.g. to machine learning, and we discuss below an example: QBoost.

1.1. 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.

Currently, the most advanced D-Wave machine is the 5,760-qubit Advantage machine with which the study King et al. [2023] was performed.

How is performance measured? It is commong to use a metric known as Time-To-Solution (TTS) when performing benchmarking studies. There, 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:

$$p_{\mathrm{TTS}} \coloneqq \frac{\#\text{ of ground state solutions}}{\#\text{ of total QA runs}}. \tag{9.1}$$

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:

$$\mathrm{TTS} = t_{\mathrm{run}} \frac{1 - \log \alpha}{1 - \log p_{\mathrm{TTS}}}. \tag{9.2}$$

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.

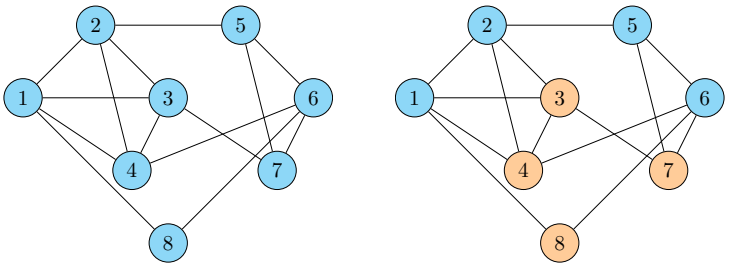
In the context of adiabatic quantum optimisation algorithms (Ising model mapping) as well as the quantum adiabatic theorem, for a problem of size N , quantum optimisers (are hoped to) solve NP-Hard combinatorial optimisation problems in time proportional to $\exp(\beta N \gamma)$ as $N \to \infty$, for positive coefficients β (scaling exponent) and γ.

It should expected that, 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.

2. More on QUBO

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

2.1. 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.



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:

$$L(x) = L_A(x) + L_B(x), \tag{9.3}$$

where

$$L_A(x) = \alpha \sum_{i=1}^{N} x_i, \tag{9.4}$$

a term that 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, and

$$L_B(x) = \beta \sum_{(u,v)\in E(G)} \frac{1 - x_u x_v}{2}, \tag{9.5}$$

a term that provides a penalty each time an edge connects vertices from different subsets. If $\beta > 0$, then we wish to minimize the number of edges between the two subsets while if $\beta < 0$ we want to maximize this number. If $\beta < 0$ is chosen, then it must be small enough so that it is never favorable to violate the other constraint L_A.

2.2. 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:

$$\begin{aligned} \max_{x\in\{0,1\}^N} \quad & cx \\ \text{s.t.} \quad & Ax = b \\ & A \in \mathbb{R}^{M\times N} \\ & b \in \mathbb{R}^M \end{aligned} \tag{9.6}$$

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

$$L(x) = L_A(x) + L_B(x), \tag{9.7}$$

where

$$L_A(x) = \alpha \sum_{j=1}^{m} \left(b_j - \sum_{i=1}^{N} A_{ij} x_i \right)^2, \tag{9.8}$$

where 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. Furthermore,

$$L_B(x) = -\beta \sum_{i=1}^{N} c_i x_i, \tag{9.9}$$

for $\beta < \alpha$ another constant. Essentially, the constants 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 there since in QUBOs (naturally) we have a minimization problem.

2.3. 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. Let N be the number of assets (things you can buy or sell in a market), μ_i the expected return of asset $i \in [N]$, σ_{ij} the covariance between the returns of asset i and asset j and R the target portfolio return. The decision variables are the weights $w_i \in \mathbb{R}$ associated to asset i for all $i \in [N]$ with.

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

$$\begin{aligned} \min_{w \in \mathbb{R}^N} \quad & \sum_{i,j=1}^{N} w_i w_j \sigma_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{N} w_i = 1, \\ & \sum_{i=1}^{N} w_i \mu_i = R. \end{aligned} \tag{9.10}$$

Intuition. Essentially, this problem amounts to the construction of an optimal portfolio from the set of all possible assets with known characteristics such as their returns, volatilities, and pairwise correlations. Realistically, we would expect to be able 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.

Quadratic programs of this form are efficiently solvable using a number of quadratic programming solvers efficiently. However, consider the case where where weights w are discrete; this situation starts resembling like a NP-Complete problem.

In such a situation Prob. (9.10) can be mapped to a QUBO suitable for QA. This is done as follows. We define the QUBO objective as:

$$L(s) = \sum_{i=1}^{N} a_i s_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} b_{ij} s_i s_j. \tag{9.11}$$

In this context

$$s_i = \begin{cases} 1 & \text{means asset } i \text{ is selected,} \\ 0 & \text{means asset } i \text{ is not selected.} \end{cases} \tag{9.12}$$

Then, given the N asset set $s = \{s_1, \ldots, s_N\}$ find the binary configuration that minimizes the $L(s)$ subject to the cardinality constraint that can be added via a penalty term $L_{\text{pen}}(s)$. Specifically, the

requirement that $\sum_{i=1}^{N} s_i = M$ is encoded via:

$$L_{\text{pen}}(s) = P\left(M - \sum_{i=1}^{N} s_i\right)^2. \tag{9.13}$$

Furthermore, in Eq. (9.12), the coefficients a_i, reflect the asset attractiveness as a standalone (think user defined hyperparameter). Assets with large expected risk-adjusted returns are commonly rewarded with negative values for a_i while assets with small expected risk-adjusted returns should be penalised with positive values of a_i. Finally, the coefficients b_{ij} reflect the pairwise diversification penalties (if positive) and rewards (if negative) and are derived from the asset pairwise correlations. For all purposes of this course, assume a_i and b_{ij} as given.

The total QUBO to be solved is:

$$\min_{s\in\{0,1\}^N} \quad L_{\text{total}}(s) \coloneqq L(s) + L_{\text{pen}}(s). \tag{9.14}$$

The minimization of this QUBO optimizes for the risk-adjusted returns by using the so-called Sharpe ration which 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.

The higher the Sharpe ratio the better returns relative to the risk taken. Volatility is usually estimated as the historical annualized standard deviation if the net asset value returns. Finally, expected returns can be either estimated as the historical returns or derived independently using e.g. Monte Carlo simulations.

3. Quantum Boost

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
$q \coloneqq (q_1, \ldots, q_N)$	vector of binary weights associated with each weak classifier

TABLE 9.1. Notation

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

$$\left(\sum_{i=1}^{N} c_i(x_t)q_i - y_t\right)^2. \tag{9.15}$$

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

$$L(s) = \sum_{t=1}^{M}\left(\sum_{i=1}^{N} c_i(x_t)s_i - y_t\right)^2 \tag{9.16}$$

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

$$\tilde{L}(s) = \sum_{t=1}^{M}\left(\sum_{i=1}^{N} c_i(x_t)q_i \sum_{j=1}^{N} c_j(x_t)s_j - 2y_t\sum_{i=1}^{N} c_i(x_t)s_i\right) + \lambda\sum_{i=1}^{N} s_i \tag{9.17}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij}q_iq_j + \sum_{i=1}^{N}(\lambda - 2C_i)s_i, \tag{9.18}$$

where

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

Remark: the penalty term added here is analogous to LASSO regression method with L_1 penalty. This is sort of ubiquitous in the ML literature. Note that ridge regression with L_2 penalty could be chosen instead.

Next, we need to map the problem to an Ising model. To do so we consider σ to be spin variables by defining

$$\sigma = 2s - 1. \tag{9.19}$$

The Ising Hamiltonian is then written as:

$$H = \frac{1}{4}\sum_{i,j=1}^{N} C_{ij}\sigma_i\sigma_j + \frac{1}{2}\sum_{i,j=1}^{N} C_i\sigma_i + \sum_{i=1}^{N}(\lambda' - C_i)\sigma_i, \tag{9.20}$$

where $\lambda' \coloneqq \frac{1}{2}\lambda$ is a rescaled penalty coefficient. QA aims to solve the problem to minimize H and compute the ground state spin configuration bit-string $|s\rangle$, with $s \in \{-1, 1\}^N$. Then, for each new sample x, the classifier is given as

$$R(x) = \sum_{i=1}^{N} s_i c_i(x) \quad \in [-1, 1]. \tag{9.21}$$

Application: QBoost

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.

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 [307,308]. This dataset consists of 30,000 samples with binary classifications:

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

There are $N = 23$ features $(F_1, \ldots, F_{23})$ that are available to extract predictive power from:

- 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)

Having these features, the next step is to construct the weak classifiers. For that each feature can be used separately as an input into a logistic regression classifier with the goal to make a binary prediction: $-1/N$ for class 0 and $+1/N$ for class 1. That, of course, would require splitting the data to a training and validation test (e.g., 0.7 training and 0.3 validation). A rule for determining the output can be set

to be a majority vote: the prediction of the (strong) classifier is given by the sum of the of the weak classifiers with values in $\{-1, +1\}$ (as required by QBoost).

It can be argued that QBoost provides an improvement on this approach by finding an optimal configuration of the weak classifiers.

Such a simple implementation can lead to results such as the ones in Fig. 9.2 (you are more than welcome to try this on your own) where QBoost shows what some people call "performance" advantage. While several, much more sophisticated classical classifiers exist, there is no fundamental reason that the same argument cannot be applied for quantum classifiers as well.

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

TABLE 9.2. Caption

4. Warm-starting QAOA

Recall, that we have discussed the Quantum Approximate Optimization Algorithm (QAOA) Farhi et al. [2014] last time. We discussed that QAOA This algorithm encodes a combinatorial optimization problem in a Hamiltonian H_C whose ground state is the optimum solution. The QAOA first creates an initial state which is the ground state of a mixer Hamiltonian H_M where a common choice is

$$H_M = -\sum_{i=1}^{N} \sigma_i^x, \tag{9.22}$$

with ground state being $|+\rangle^{\otimes n}$. Then, recall, for depth-L QAOA, we apply L times the unitary $U_{\text{QAOA}} = U_C(\gamma)U_B(\beta)$ defined as:

$$U_C(\gamma) \coloneqq e^{-\imath\gamma H_C} \tag{9.23}$$

$$U_B(\beta) \coloneqq e^{-\imath\beta H_M}. \tag{9.24}$$

The result is:

$$U_{\text{QAOA}}|+\rangle^{\otimes n} = |\gamma, \beta\rangle. \tag{9.25}$$

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.

While a very promising algorithm, initially it lacked any 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 (which for MAXCUT 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).

While several improvements of the QAOA have been developed in the literature, we will focus here on warm-starting QAOA Egger et al. [2021].

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

$$\min_{x\in\{0,1\}^n} x^T Q x + \mu^T x. \tag{9.26}$$

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, and in the following, we only add μ when it simplifies the notation in the given context. Note that, as discussed, practically any mixed-integer linear program can be encoded in a QUBO it is automatically NP-Hard.

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

$$\min_{x\in[0,1]^n} x^T Q x \tag{9.27}$$

is a convex quadratic program and the optimal solution c^* of the continuous relaxation is easily obtainable with classical optimizers. However, if Q is not positive semidefinite (and in many applications of interest there is no reason to be) one can obtain another continuous relaxation, the semidefinite program (SDP):

$$\begin{aligned} \max_{Y\in\mathbb{S}^n} \operatorname{tr}(QY) \\ \operatorname{diag}(Y) = \mathbf{1} \\ Y \succeq 0, \end{aligned} \tag{9.28}$$

where $\mathbb{S}^{n\times n}$ denotes the set of $n\times n$ symmetric matrices, $\mathbf{1}$ is an n-vector of ones, and $Y\succeq 0$ denotes that Y must be positive semidefinite. Given the optimal solution Y^* to the SDP above, there exist several approaches to generating solutions[1] of the corresponding (QUBO), often with approximation guarantees. Furthermore, quantum computers offer the prospect of some speed-ups in solving SDPs (not discussed in these lectures).

Warm-starting QAOA. The solutions of either continuous-valued relaxation discussed above can be used to initialize VQAs: this is known as warmstarting them Gondzio [1998]. Let us focus on how to warm-start the QAOA Farhi et al. [2014].

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. Then, after the procedure described initially, that is utilizing the unitary U_{QAOA}, one performs the final measurement which can be considered as a randomized rounding. Warm-starting amounts to replacing the initial equal superposition state $|+\rangle^{\otimes n}$ with a state

$$|\phi^*\rangle = \bigotimes_{i=0}^{n-1} \hat{R}_y(\theta_i)\,|0\rangle_n \tag{9.29}$$

which corresponds to the solution c^* of the relaxed Problem (9.27). Here, $\hat{R}_Y(\theta_i)$ is a θ_i-parametrized rotation around the y-axis of the qubit (see Fig. 9.1) and $\theta_i \coloneqq 2\arcsin(\sqrt{c_i^*})$ for c_i^* given as the solution of QUBO (9.27).

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

$$H_M^{\mathrm{ws}} = \sum_{i=1}^{n} H_{M,i}^{\mathrm{ws}} \tag{9.30}$$

where

$$H_{M,i}^{\mathrm{ws}} = \begin{pmatrix} 2c_i^* - 1 & -2\sqrt{c_i^*(1-c_i^*)} \\ -2\sqrt{c_i^*(1-c_i^*)} & 1-2c_i^* \end{pmatrix} \tag{9.31}$$

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 $U_M(\beta) = e^{-\imath\beta H_M^{\mathrm{ws}}}$.

For technical reasons Egger et al. [2021] one has to actually modify the definition of θ_i as

$$\begin{aligned} \theta_i &= 2\arcsin\left(\sqrt{c_i^*}\right) && \text{if} \quad c_i^* \in [\varepsilon, 1-\varepsilon] \\ \theta_i &= 2\arcsin(\sqrt{\varepsilon}) && \text{if} \quad c_i^* \le \varepsilon \\ \theta_i &= 2\arcsin(\sqrt{1-\varepsilon}) && \text{if} \quad c_i^* \ge 1-\varepsilon. \end{aligned}$$

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.

[1] As a matter of facr, a classical laptop can solve instances of (SDP) relaxations of QUBO, where Q has 10^{13}.

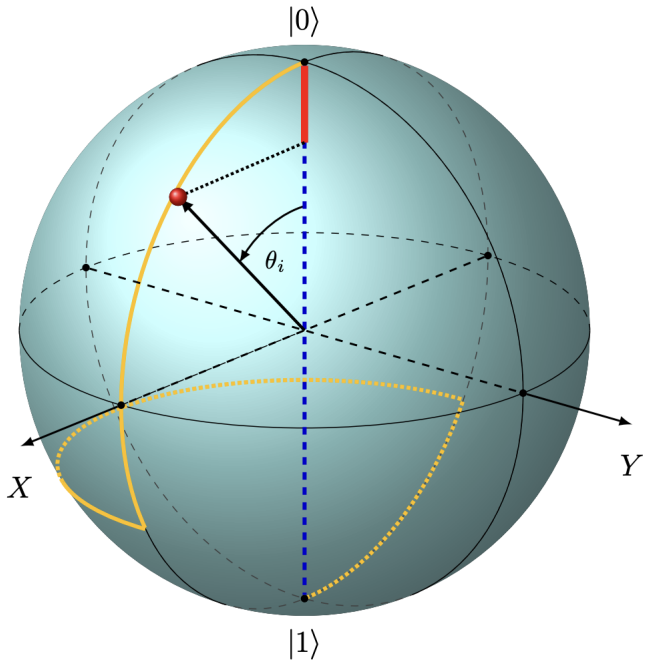


FIGURE 9.1. The initial state is given by the red segment. The yellow path shows the evolution of the quantum state starting at $|0\rangle$ to $R_y(-\theta_i)R_z(-\beta)R_y(2\theta_i)$ for $\beta = \pi/2$.

This directly follows from (surprise) the adiabatic theorem and the fact that we start in the ground state of the mixer which overlaps with all computational basis states including the optimal solution.

For large enough L, WS-QAOA therefore the adiabatic evolution transforming the ground state of the mixer into the ground state of 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.

Several other variations of WS-QAOA have been studied, for example "rounded warm-started" QAOA. Such a technique is very appealing for application on NISQ devices for combinatorial optimization problems.

Below we quote a nice experimental demonstration from Egger et al. [2021].

There, the authors investigate the role of the warm-start mixer operator $\hat{H}_M^{(ws)}$ by replacing it with the standard mixer $-\sum_{i=0}^{n-1} \hat{X}_i$ while using the initial state given by the continuous solution c^*. Under these conditions the energy of the optimized state does not converge to the minimum energy, see blue triangles in Fig. 9.2(b). The probability of sampling the optimal discrete solution is between warm-start and standard QAOA but depends heavily on the initial point given to COBYLA, see Fig. 9.2(a). These results further justify the use of the modified mixer in WS-QAOA.

They even proceed to further illustrate the advantage of a warm-starting QAOA at low depth by solving 250 random portfolio instances with warm-started and standard QAOA, both at depth $L = 1$. There the standard QAOA produces variational states that poorly approximate the ground state, see the histogram of E^*_{cold} in Fig. 9.3(a). However, WS-QAOA produces optimized variational states that are much closer to the minimum energy of each problem Hamiltonian. Furthermore, we find that WS-QAOA tends to produce better solutions when the overlap $d^{*T}c^*/B$ between the optimal solutions to the discrete and relaxed problems is closer to 1.

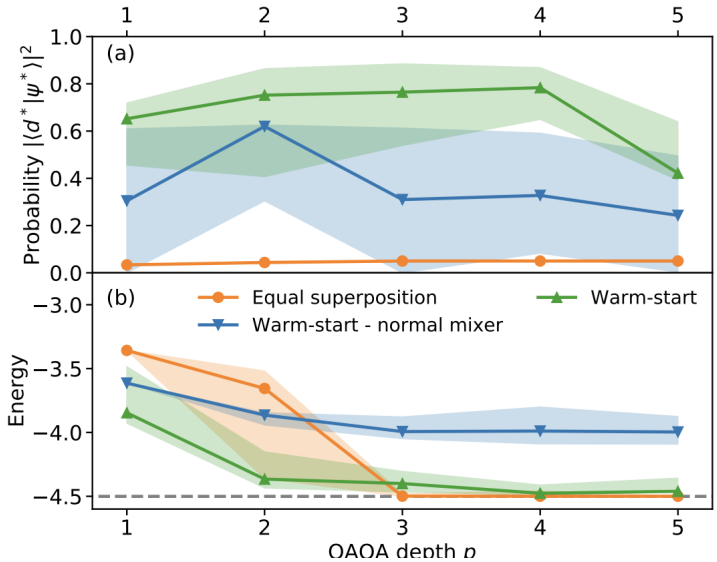


FIGURE 9.2. (a) Probability to sample the optimal state $|d^*\rangle$ from the optimized trial state $|\psi^*\rangle$ and (b) energy of $|\psi^*\rangle$ for warm-start and standard QAOA at different depths for $n = 6$ assets and $q = 2$. The optimal discrete and continuous solutions are $d^* = (0, 0, 1, 1, 1, 0)$ and $c^* \simeq (0.17, 0, 0.97, 0.73, 1.0, 0.14)$, respectively. QAOA is run ten times with different initial random guesses for (β, γ) chosen uniformly from $\pm 2\pi$. The thick lines show the median of the ten runs while the shaded areas indicate the 25% and 75% quantiles. The gray dashed line shows E_0.

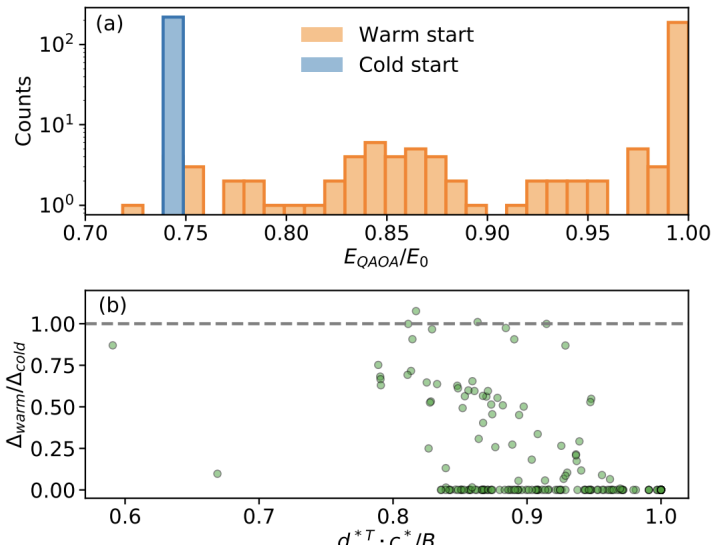


FIGURE 9.3. Improvement of depth-one WS-QAOA over standard QAOA for 250 random portfolio instances with $q = 2$ (q controls the risk-return trade-off).

5. Asset Management and Monte Carlo Simulations

Derivative pricing using a quantum computer.

What are derivatives and why one would be interested in using a quantum computer?

Short answer: derivatives are financial instruments that make some people rich and quantum computers can do things (in principle) quadratically faster. Use this notebook.

Bibliography

Daniel J Egger, Jakub Mareček, and Stefan Woerner. Warm-starting quantum optimization. *Quantum*, 5:479, 2021.

Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

Jacek Gondzio. Warm start of the primal-dual method applied in the cutting-plane scheme. *Mathematical Programming*, 83(1-3):125–143, January 1998. doi: 10.1007/bf02680554. URL `https://doi.org/10.1007/bf02680554`.

Andrew D King, Jack Raymond, Trevor Lanting, Richard Harris, Alex Zucca, Fabio Altomare, Andrew J Berkley, Kelly Boothby, Sara Ejtemaee, Colin Enderud, et al. Quantum critical dynamics in a 5,000-qubit programmable spin glass. *Nature*, pages 1–6, 2023.