Hybrid quantum-classical optimization for financial index tracking

Samuel Fernández-Lorenzo

BBVA Client Solutions Research & Patents, Calle Sauceda 28, 28050 Madrid, Spain.

E-mail: samuel.fernandez.lorenzo.contractor@bbva.com

Diego Porras

Instituto de Física Fundamental, IFF-CSIC, Calle Serrano 113b, 28006 Madrid, Spain.

Juan José García-Ripoll

Instituto de Física Fundamental, IFF-CSIC, Calle Serrano 113b, 28006 Madrid, Spain.

Abstract. Tracking a financial index boils down to replicating its trajectory of returns for a well-defined time span by investing in a weighted subset of the securities included in the benchmark. Picking the optimal combination of assets becomes a challenging NP-hard problem even for moderately large indices consisting of dozens or hundreds of assets, thereby requiring heuristic methods to find approximate solutions. Hybrid quantum-classical optimization with variational gate-based quantum circuits arises as a plausible method to improve performance of current schemes. In this work we introduce a heuristic pruning algorithm to find weighted combinations of assets subject to cardinality constraints. We further consider different strategies to respect such constraints and compare the performance of relevant quantum ansätze and classical optimizers through numerical simulations.

1. Introduction

Many relevant problems in quantitative finance translate into daunting computational tasks, such as combinatorial optimization problems and Monte Carlo simulations [1], suffering from lack of parallelization or slow convergence. The field of Quantum finance is an emergent branch of quantum physics and quantum computing that develops new algorithms and formulations of financial problems, to address these problems and limitations [2]. Recent works in quantum finance have explored the design of optimal trading trajectories [3], credit scoring [4], portfolio optimization [5, 6, 7, 8], Monte Carlo pricing of derivatives [9], risk analysis [10] or financial crisis forecast [11], among others.

While financial applications show promising results and trends for NISQ computations, they also highlight the bottleneck imposed by processor sizes and decoherence. In such a framework of hybrid quantum-classical computation, it is equally important to benchmark and fine tune existing algorithms, as well as seeking new algorithmic simplifications or strategies that provide useful applications in resource-limited environments. In this work we address both questions, using as model the problem of *index tracking*.

Aggressive investors or hedge funds seek to beat the market performance by frequently trading based on expectations regarding macroeconomic and company-specific developments, usually referred as active investment; however, only a small fraction of hedge funds are successful in doing so. More risk averse investors focus on passive investment, using financial instruments—funds, ETFs, derivatives—which imitate the performance of a benchmark index, such as the S&P 500. Interestingly, the emulation of indices is not constrained to long-term passive investment. Synthetic indices must be reproduced also when providing counterpart risk in derivative trading—e.g. S&P 500, EuroStoxx and other indices can be traded as futures—, and appear in other markets, such as commodities. In all these scenarios, the design of the *index tracking* instrument itself is a complex task, which can be improved by having faster and more accurate algorithms. In general, one can think of this problem as reducing transaction costs in portfolio investments, which is of wide interest for obvious reasons.

In this paper we explore a formulation of the index tracking problem as a mixed-integer optimization problem, where a benchmark index is imitated by a basket of d assets in a universe with N possible instruments, by selecting and fine tuning the weights of those assets in the basket. Note that the selection of d assets is already a combinatorial optimization problem that explodes as $\binom{N}{d} \sim N^d$. Exploring all configurations by brute force quickly becomes unfeasible for indices composed by dozens of instruments, so approximation methods [12, 13] and heuristic approaches come into play [14, 15, 16, 17, 18, 19]. However, if we want to explore the optimization power of NISQ computers, we are faced with the overhead imposed by encoding multiple assets with finite precision. This severely limits the size and interest of problems that can be addressed with existing hardware.

In this work we address the problem of index tracking using a new hybrid

quantum-classical algorithm that we call *heuristic pruning*. Our method iteratively nests two interdependent optimization problems: a classical convex optimization phase to determine the relative weights of the assets, combined with a quadratic binary optimization that determines the relevant subset of securities that reproduces the index. This approach makes best use of small Noisy Intermediate Scale Quantum (NISQ) devices [20], maximizing the number of assets that are described by a single quantum register—i.e. if we have N qubits we can track indices with up to N securities. Conveniently enough, our pruning algorithm can easily be adapted to solve other relevant problems in industry, like selecting variables in a multi-variate linear regression. We demonstrate that the pruning method can solved in NISQ quantum computers (e.g. IBM-Q), using variational methods with a very small number of gates but very good performance. However, the same method can be extended to work with quantum annealers (D-Wave's) or quantum-inspired optimizers. Finally, the formulation is sufficiently flexible that it can readily incorporate other constraints appearing in realistic scenarios, like capital concentration constraints, even though we do not consider them in this work.

Section 2 enunciates our index tracking problem. In section 3 we describe and analyze the pruning algorithm. In Sec. 4 we study ways to include hard cardinality constraints by either the mathematical encoding or the quantum circuit, comparing both methods through numerical simulations using real intra-day data of the PHLX Oil Service Sector (OSX) index, composed of 15 companies involved in the oil services sector (cf. 1). By relaxing the previous hard constraints and allowing final portfolio sizes not imposed *a priori*, we find a more gentle optimization problem for the variational parameters in section 5. Finally, Sec. 6 discusses the conclusions of our study and next steps.

2. Index-tracking problem

The goal of thee index tracking problem is to replicate, over a given look-back window, a benchmark index by selecting a portfolio of d components out of a large universe $N \ll d$ of possible assets. We expect that the smaller tracking portfolio will closely match the returns of the benchmark index, while at the same time reducing the transaction costs and simplifying the management process required to keep both the portfolio in-sync with the index.

Let us assume that we know the prices $P_j(t)$ of the assets in our universe over an interval of time $t \in [0, T]$, which we divide into periods with fixed size $t_n = n \times \Delta t$. At any time, the composition of our portfolio is determined by a set $c_j(t)$ of product units that are adapted during the tracking period. The value of the portfolio during this interval is

$$P(t) = \sum_{j=1}^{d} c_j P_j(t), \quad t \in [0, T]. \tag{1}$$

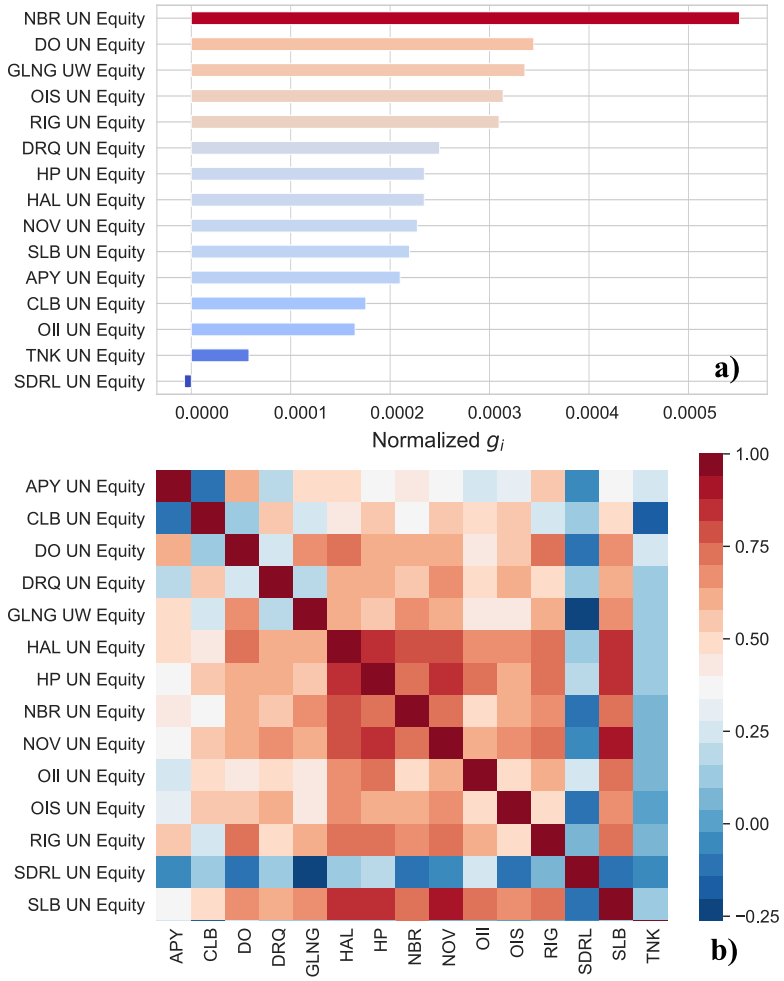

Figure 1. Composition of the PHLX Oil Service Sector (OSX) index, with look-back window on 2019-06-17. (a) Names of the assets and weights on the normalized linear components g_i. (b) Correlation heatmap between equities, related to the quadratic form Σ_{ij}.

Within each sub-interval, we can define the overall return of the portfolio $r_p(t_n)$ as the relative increment in portfolio value. It can be related to the returns of individual assets $r_i(t_n) = P_i(t_n + \Delta t)/P(t_n) - 1$

$$r_p(t_n) = \frac{P(t_n + \Delta t) - P(t_n)}{P(t_n)} = \sum_j^N w_j r_j(t_n) = \mathbf{w}^T \mathbf{r}(t_n), \tag{2}$$

through a vector of weights $\mathbf{w} \in \mathbb{R}^N$ characterizing the portfolio

$$w_j = \frac{c_j(t) P_j(t)}{\sum_j^n c_j(t) P_j(t)} \in [0, 1]. \tag{3}$$

The benchmark index $I(t)$ is known through a time series of returns, $r_I(t_n) = I(t_n + \Delta t)/I(t_n) - 1$, taken by consecutive snapshots at periodic intervals. We will use this information, to construct the portfolio with d assets (and fixed weights) that best replicates the index' trajectory [17]. Our assumption is that this optimal portfolio will

accurately follow the index over a short time horizon, $t \in [T, T+T_H]$. After this time, a new optimal portfolio will be computed, leading to a reevaluation of the weights w_j that can be achieved through buying or selling some product units c_j.

To implement the search of the optimal portfolio, we introduce a tracking error T_{err} that measures the *distance* between the sequence of returns of the index and our portfolio. We use the mean squared error between both time series, a quadratic form

$$T_{\text{err}} = \sum_{n}^{T} (r_p(t_n) - r_I(t_n))^2 = \sum_{i,j=1}^{N} w_i \Sigma_{ij} w_j - 2\sum_{j=1}^{N} w_j g_j + \epsilon_0, \tag{4}$$

with a matrix Σ_{ij}, a vector g_i and an offset ϵ_0, defined as

$$\Sigma_{ij} = \sum_{n=0}^{n_T} r_i(t_n) r_j(t_n), \quad g_j = \sum_{n=0}^{n_T} r_j(t_n) r_I(t_n), \quad \epsilon_0 = \sum_{n=0}^{n_T} r_I(t_n)^2. \tag{5}$$

The index can also be expressed in terms of a vector of weights $\mathbf{W} \in \mathbb{R}^N$ valid for the specified time interval‡, giving us explicit formulas for the components as

$$g_i = \sum_{j=1}^{N} \Sigma_{ij} W_j \quad \epsilon_0 = \sum_{i,j=1}^{N} W_i \Sigma_{ij} W_j, \tag{6}$$

The search for the optimal portfolio is a mixed-integer optimization problem with cost function (4), minimized with respect to the weights $\mathbf{w}$ and subject to cardinality and normalization constraints,

$$\underset{\mathbf{w}}{\text{argmin}}\, T_{\text{err}} := \underset{\mathbf{w}}{\text{argmin}} \left[\mathbf{w}^T \Sigma \mathbf{w} - 2\mathbf{w}^T \mathbf{g} + \epsilon_0\right] \tag{7}$$

$$||\mathbf{w}||_0 = d, \tag{8}$$

$$||\mathbf{w}||_1 = 1, \tag{9}$$

$$w_j \geq 0 \quad \forall j. \tag{10}$$

Here, $||w||_0 = \sum_j |w_j|^0$ and $||w_j||_1 = \sum_j |w_j|$ stand for the 0-norm and the 1-norm respectively. Eq. (10) is introduced to avoid short selling, and the integer nature of the problem is manifested when selecting a subset of d assets so that Eq. (8) is respected. This formulation assumes that $\mathbf{w}$ is a vector of continuous values, which is a good approximation commonly employed in practice. However, one could also tackle the problem as fully combinatorial recalling that equities are bought and sold in fixed lot sizes, thereby requiring a discretization in the values of w_j as discussed in [21].

Notice that the symmetric matrix Σ_{ij} is similar to a covariance matrix among assets, whereas the vector g_j accounts for covariance-like components with respect to the benchmark. Hence, the cost function (7) can be interpreted similarly to a Markowitz's porftfolio optimization: the mission of the quadratic term is to reduce the volatility associated to the uncertain evolution of the assets while the linear term favors those assets more correlated with the benchmark [22, 23].

‡ Even if those weights are not explicitly provided or available in financial data, such vector can be computed by convex optimization of the universe of N index assets with no other restriction that demanding positive weights for all of them and proper normalization (budget constraint).

The problem, as stated, tends to favor assets with larger weights in the index. In realistic scenarios there may be further constraints to ensure a well-diversified portfolio, such as a capital concentration constraint of the form,

$$\mathbf{l} \leq \mathbf{C}\mathbf{w} \leq \mathbf{u}, \tag{11}$$

where $\mathbf{l}$ and $\mathbf{u}$ are vectors indicating the lower and upper bounds of certain linear combinations of investments gathered in the matrix $\mathbf{C}$.

For simplicity we shall not consider this sort of constraint. Instead, we focus on drawing the quantities Σ_{ij} and g_i from a data set that leads to computationally demanding problem. More precisely, we will look for indices with assets that are strongly correlated, in a way that makes the quadratic terms are comparable to the linear terms. Otherwise, the problem can be solved in good approximation just by sorting assets by their contribution to the index and picking the top d from the list. For our numerical studies, we picked an index consisting of a cluster of 15 oil companies: the PHLX Oil Service Sector (OSX) index. The dataset gathers prices of all the instruments with interval Δt of 20 minutes over several days. We take look-back windows corresponding to a single day, and average over several days. An example of the correlation heat map of these equities and their corresponding (normalized) component g_i for a random day (2019-06-17) is shown in figure 1.

3. Pruning algorithm

3.1. One-step pruning

Index tracking is a mixed-integer optimization that implicitly joins a computationally hard task—the selection of assets that are used to imitate the index, within an exponentially large family of choices—, with a classically tractable, convex optimization problem—assigning the weights to the selected assets—. This suggest that we split index tracking into two problems, which will later addressed by a quantum and a classical computer, in line with the current philosophy of hybrid classical-quantum computations.

Let us define a vector $\mathbf{x}$ of decision variables which determine whether the $i-$th asset is included in the portfolio ($x_i = 1$) or omitted ($x_i = 0$). A first iteration of our idea is the *single-step selection algorithm* (1-SA). We solve the combinatorial optimization problem of selecting a portfolio of d assets

$$\begin{aligned} &\underset{\mathbf{x}}{\operatorname{argmin}}\, S_{\text{err}} := \underset{\mathbf{x}}{\operatorname{argmin}} \left[\mathbf{x}^T \Sigma \mathbf{x} - 2\mathbf{x}^T \mathbf{g}\right] \\ &||\mathbf{x}||_0 = d. \end{aligned} \tag{12}$$

This selection is used to build an index tracking problem with smaller matrices, $\Sigma_r(\mathbf{x})$ and $g_r(\mathbf{x})$, that only contain the rows and columns where $x_i \neq 0$. The reduced problem gives us a smaller vector of weights $\mathbf{w}_r \in \mathbb{R}^d$

$$\begin{aligned} &\underset{\mathbf{w}_r}{\operatorname{argmin}}\, T^r_{\text{err}}(\mathbf{x}) := \underset{\mathbf{w}_r}{\operatorname{argmin}} \left[\mathbf{w}_r^T \Sigma_r(\mathbf{x}) \mathbf{w}_r - 2\mathbf{w}_r^T \mathbf{g}_r(\mathbf{x}) + \epsilon_0\right] \\ &||\mathbf{w}_r||_1 = 1, \text{ and } w_{rj} \geq 0 \quad \forall j. \end{aligned} \tag{13}$$

In practice, this formulation is problematic, because the selection of assets does not incorporate information about the relative importance of securities in the final weights. A better strategy is to reverse the previous sequence, creating the *single-step pruning algorithm* (1-PA). In this version, the cost function of the selection problem incorporates the weights, through the transformation $x_i \to w_i x_i$. More precisely, we first solve the convex optimization of the whole index

$$\begin{aligned} &\underset{\mathbf{w}}{\operatorname{argmin}}\, T_{\text{err}}^{full} := \underset{\mathbf{w}}{\operatorname{argmin}} \left[\mathbf{w}^T \Sigma \mathbf{w} - 2\mathbf{w}^T \mathbf{g} + \epsilon_0\right] \\ &||\mathbf{w}||_1 = 1, \text{ and } w_j \geq 0 \quad \forall j. \end{aligned} \tag{14}$$

This information is incorporated into a diagonal matrix $D_{ij}(w) = w_i \delta_{ij}$, with which we select the assets in a modified combinatorial optimization problem

$$\begin{aligned} &\underset{\mathbf{x}}{\operatorname{argmin}}\, P_{\text{err}}(\mathbf{w}) := \underset{\mathbf{x}}{\operatorname{argmin}} \left[\mathbf{x}^T \mathbf{D}(\mathbf{w}) \Sigma \mathbf{D}(\mathbf{w}) \mathbf{x} - 2\mathbf{x}^T \mathbf{D}(\mathbf{w}) \mathbf{g}\right] \\ &||\mathbf{x}||_0 = d. \end{aligned} \tag{15}$$

Finally, we update the optimal weights for this combination through another convex optimization problem on a reduced subspace like in equation (13).

3.2. k-step pruning

We can use using 1-PA as the basis for an *iterative pruning algorithm* that constructs solutions for decreasing portfolio sizes. More precisely one would design a discrete schedule with k steps and decreasing universe sizes $N_1 = N > d_1 = N_2 > \ldots \geq N_k > d_k = d$. On the k-th step, the 1-PA algorithm searchs a portfolio of size d_k that approximates a new index (15) defined over a universe with N_k assets, with smaller matrices Σ_r and g_r constructed for the assets in that universe. One useful schedule is to linearly decrease the sizes by a fixed amount $N_i = N - s \times i$, but other possibilities are feasible.

We can combine the iterative pruning with stochastic optimization methods, such as a quantum optimization algorithm, to solve the 1-PA steps. In this scenario, we have to fine tune not only the universe size, but also the number of repetitions of the stochastic method. We suggest to gradually increase the number of repetitions as the universe size shrinks, according to the schedule $r \to r_0 + \alpha r$. Heuristically, in early stages where $d \sim N$, we are more likely to retain assets in the optimal portfolio, so it makes sense to spend less repetitions. In later stages, each repetition will be exponentially cheaper and more accurate, due to the decrease in universe size, and it pays off to use the same number of resources to get more accurate intermediate solutions. We refer to the combination of universe size and repetition schedule as k-PA. It is described in Algorithm 1 as a pseudocode function to calculate a target portfolio of size d_{target} and weights w_j from and index of size N in steps of size s.

Algorithm 1 Iterative pruning algorithm

Input: $N, d_{\text{target}}, r_0, \alpha, s$
  $d \leftarrow N$
  get ideal weights $w_j = W_j$ or find them from $\text{argmin}_{\mathbf{w}} T_{\text{err}}^{full}$
  while True **do**
    $r \leftarrow r_0 + \alpha r$
    $d \leftarrow \max(p, d - s)$
    for 1:r **do**
      approximately solve $\text{argmin}_{\mathbf{x}} P_{\text{err}}$ with weights w_j, to select d assets
      keep the best solution $\mathbf{x}$
    end for
    solve $\text{argmin}_{\mathbf{w}} T_{\text{err}}^{r}(\mathbf{x})$ for selected assets in $\mathbf{x}$ to obtain new weights w_j
    construct $\Sigma_r(\mathbf{x})$ and $g_r(\mathbf{x})$
    $N \leftarrow d$
    if $N = d_{\text{target}}$ **then**
      break
    end if
  end while

3.3. Performance analysis

We have compared the performance of the 1-SA and 1-PA algorithms against the search for the optimal solution of the full index tracking problem, without any simplification. Figures 2a-b compare the error of the simplified algorithms with the optimal solution, for all ratios of tracker size d vs universe size N. Points located on the graph diagonal represent scenarios where the simplified algorithm finds the optimal optimal solution. The concentration of points along the diagonal is an evidence of the quality of the 1-PA method, which offers a much better heuristic than the 1-SA. We quantify the difference using the Pearson's correlation coefficient, which is 0.92 and 0.68 for 1-PA and 1-SA, respectively. Another way to compare these methods is to study the probability that an algorithm finds a portfolio with relative error with respect to the optimal one $\Delta = (T_{\text{err}} - T_{\text{err}}^{\text{full}})/T_{\text{err}}^{\text{full}}$. As shown in figure 2c, 1-PA achieves an error $\Delta \leq 20\%$ in 62.5% of the runs, while 1-SA only does it with 17.7% probability.

Our analysis shows that solving (15) offers a satisfactory heuristics for finding the optimal or close to optimal portfolio for any given size d. Motivated by this, we have studied also the performance of the k-PA method, in combination with classical simulated annealing, a well-known classical stochastic algorithm. As publicly available reference, we used the dimod library [24] for Quadratic Unconstrained Binary Optimization (QUBO) problems, without fine tuning, but solving the problem with the hard-constraint penalty described in the next section. For a fair comparison, the experiment is designed to arrive to the same portfolio size ($d = 5$) with the same number of total overall repetitions (120) through 1-PA ($r_0 = 120, \alpha = 0$), 2-PA ($r_0 = 120, \alpha = 4$),

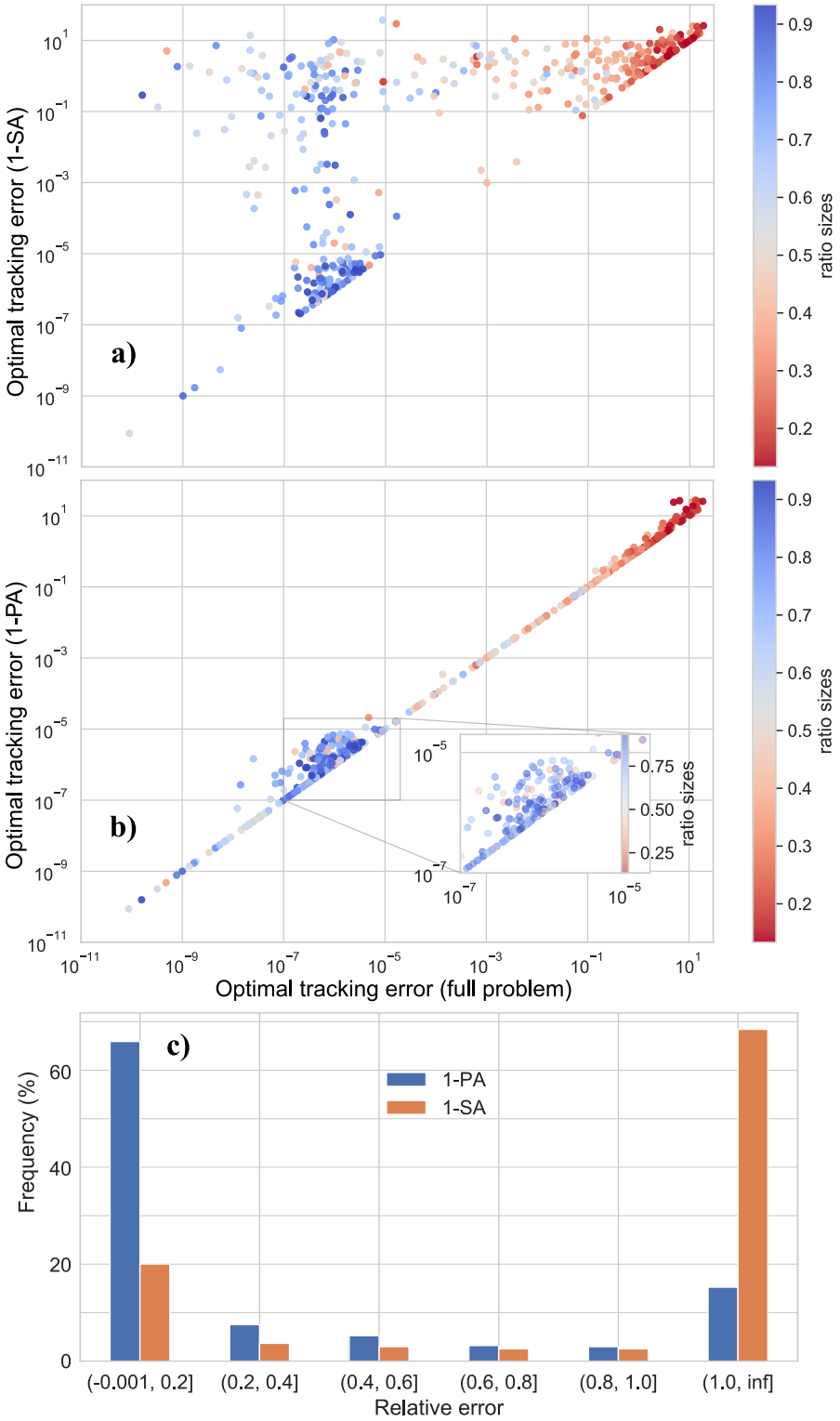

Figure 2. Single-step selection and pruning algorithms. Global minimum error in (a) 1-SA and (b) 1-PA vs. the optimal solution of the full problem, for random days and ratios d/N. (c) Probabilities of achieving a certain relative error in 1-SA and 1-PA, compared to the optimal solution.

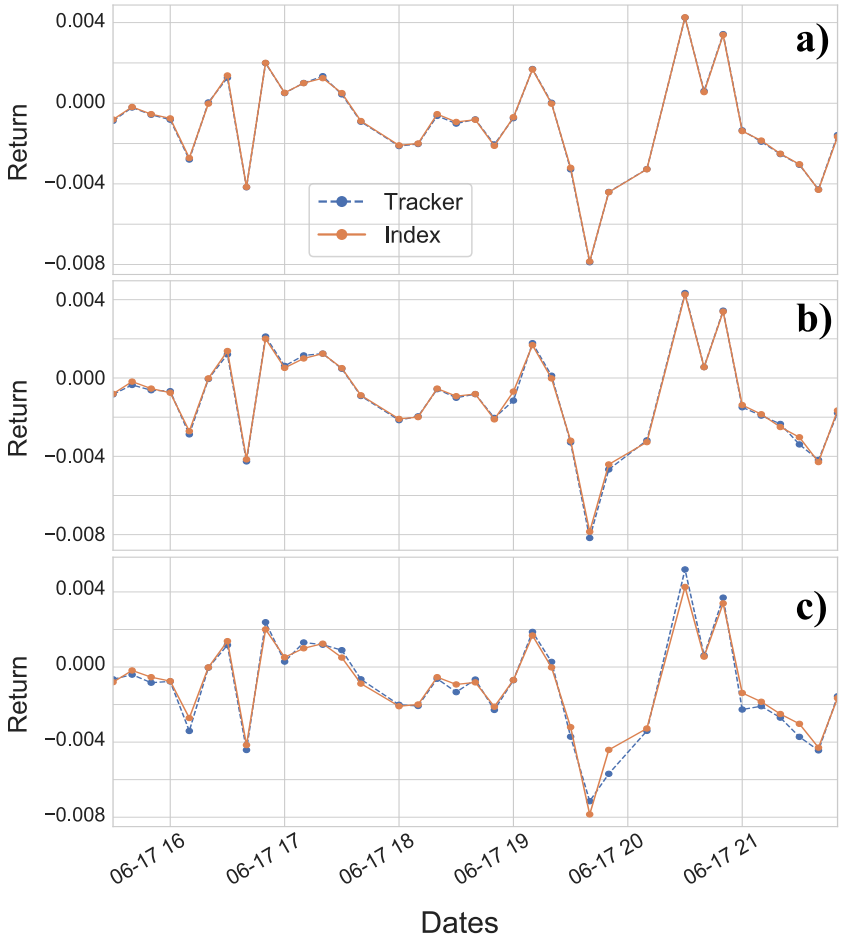

Figure 3. Example of 3-step pruning algorithm with simulated annealing for OSX index. Benchmark in solid orange line, tracker in dotted blue line. a) First step, reduction from 15 assets to 11. b) Second step, reduction from 11 assets to 7. c) Third step, reduction from 7 assets to 5. Final tracking error 1.3%.

and 3-PA ($r_0 = 20, \alpha = 1$) respectively.

The series of figures 3 illustrates an example of the 3-PA sequence for a random day (2016-06-17), with a final relative error of 1.3%. The bar plot in figure 4a displays a comparison of the median of the relative error for the multi-step pruning algorithm with the optimal of the full problem, while the boxplot 4b compares the time spent in each algorithm. Our numerical experiment indicates that the k-PA is not only faster as we increase the number of steps, but also more accurate.

The resulting algorithm offers an alternative appealing heuristic with respect to others local-search algorithms studied in the literature, like the threshold accepting method [18, 25].

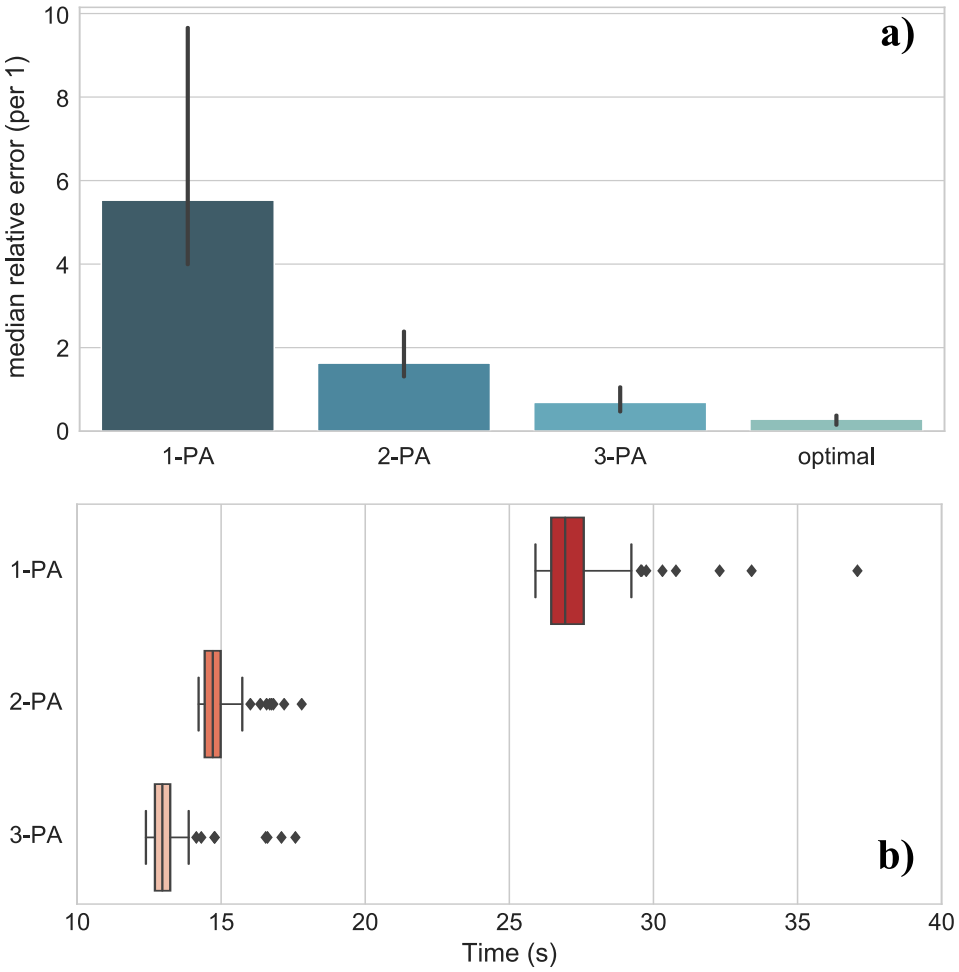

Figure 4. Comparison of multi-step pruning algorithm with simulated annealing. a) Bar plot of the median of relative errors for 1,2,3-step pruning algorithm and global minimum obtained through diagonalization. b) Box plot of the time consumed in each trial.

4. Hard-constraint formulations

4.1. Quantum variational algorithms

Having established that the pruning algorithm offers accurate solutions the index tracking problem, we shall now address how to solve the integer optimization problem from Eq. (15) using variational quantum circuits for combinatorial optimization [26, 27, 28, 29, 30, 31]. Generally speaking, these methods represent the solution of the optimization problem as a quantum superposition over all possible configurations. The wavefunction is parameterized by the angles and phases of the quantum gates used to build it. These real parameters are processed by some classical optimizer, in a hybrid classical-quantum model that starts from rather uniform superpositions and aims at concentrating probability on the best arguments to our cost function. The use of quantum variational methods opens new questions, such as the shape and structure of the ansatz, the type of classical optimizers to use, or the role of quantum fluctuations

and entanglement in exploring the complex and noisy space of solutions.

Our first task is to encode the quantum optimization problem defined by (15) in a quantum processor. Quantum variational algorithms where original devised for tackling unconstrained optimization problems. One alternative to circumvent this limitation is to include the constraints (15) into the QUBO cost function, as quadratic penalties.

$$\underset{\mathbf{x}}{\operatorname{argmin}} P_{\mathrm{err}}^{QUBO} = \underset{\mathbf{x}}{\operatorname{argmin}} \mathbf{x}^T \mathbf{D}\Sigma\mathbf{D}\mathbf{x} - 2\mathbf{x}^T\mathbf{D}\mathbf{g} + P\left(\mathbf{1}^T\mathbf{x} - d\right)^2. \tag{16}$$

Here, $\mathbf{1}$ is a vector of ones and P is a penalty weight—large enough to prevent unfeasible configurations, but small enough to allow tunneling between different portfolios that satisfy the constraints. The optimization problem (16) can be brought to a quadratic form $P_{\mathrm{err}}^{QUBO} \sim \mathbf{x}^T Q \mathbf{x}$, including the linear terms§.

We can write the QUBO cost function as an Ising model,

$$H_{\mathrm{err}} = \sum_{ij} J_{ij}\sigma_i^z\sigma_j^z + \sum_i \sigma_i^z h_i + E_0, \tag{17}$$

by means of the mapping $x_i \rightarrow \frac{1}{2}(1-\sigma_i^z)$, from bits to Pauli Z operator σ_i^z. Our optimization problem translates to finding one of the ground states $|s_1, s_2, \ldots, s_N\rangle$ of H_{err}. We will approximate this as a variational method

$$\underset{\vec{\theta}}{\operatorname{argmin}} \langle\Psi(\vec{\theta})|H_{\mathrm{err}}|\Psi(\vec{\theta})\rangle \tag{18}$$

finding the best choice $\vec{\theta}$ over a family of wavefunctions $|\Psi(\theta)\rangle$, constructed with gates that are parameterized by the angles θ_i. The optimization is done classically, constructing the wavefunction in the quantum computer, estimating the energy and estimating an update for the parameters. We expect that, if the variational family is dense enough, it will approximate the ground state with arbitrary accuracy, or at least get close to it.

4.2. Variational states

Given the limitations of NISQ devices concerning circuit depth, a first approach is to restrict ourselves to hardware-efficient trial states [28], variational wavefunctions created within the limitations of existing devices. More precisely, we will use the VQE Ry ansatz (henceforth just VQE) introduced in Ref. [32]. Formally, the wavefunction reads

$$|\Psi_{VQE}(\vec{\theta})\rangle = \left(\prod_{j=2}^{p} \exp\left(-i\sum_l \theta_{jl}\sigma_i^y\right) U_{ent}\right) \exp\left(-i\sum_l \theta_{1l}\sigma_i^y\right)|\Psi_0\rangle\,. \tag{19}$$

The ansatz starts from the zero state of the quantum register $|\Psi_0\rangle = |0\rangle_1 \cdots |0\rangle_N$, and then subsequent layers of local rotations and entangling operations are applied. The variational parameters θ_{jl} determine the local rotation on the l-th qubit in the j-th layer. As entangling operations we use a sequence of control-Z gates between nearest-neighbor qubits, assuming the qubits form a 1D register.

§ Note that for binary variables $x_i^2 = x_i x_i = x_i$ and $\sim \mathbf{b}^t\mathbf{x} = \sum_{ij} x_i x_j b_i \delta_{ij}$.

The Quantum Approximate Optimization Algorithm [26] (QAOA) is more hardware-demanding ansatz, that implements Trotterized version of quantum annealing. The variational parameters are the angles β_n of global x-rotations and of γ_n entangling operations implemented by the problem Hamiltonian,

$$|\Psi_{QAOA}(\vec{\gamma},\vec{\beta})\rangle = \prod_{n=1}^{p} \exp\left(-i\beta_n \sum_i \sigma_i^x\right) \exp\left(-i\gamma_n H_{\text{err}}\right) |\Psi_+\rangle \,. \tag{20}$$

The initial state is the uniform superposition of all register states $|\Psi_+\rangle = \bigotimes_{j=1}^{N} |+\rangle_j$, with $|+\rangle_j = (1/\sqrt{2})(|0\rangle_j + |1\rangle_j)$. In the limit of very large number of layers, QAOA may converge to the dynamics of a quantum annealing process and therefore prepare the ground state with high fidelity. In practice, the number of layers p is smaller, but the ansatz captures the structure of the problem, which is why it is expected to perform better than the generic VQE method, but with an added cost of more gates to implement H_{err}.

Another strategy is to force the quantum circuit to search over subspaces of the Hilbert spaces that respect the cardinality constraint without the need of imposing an energy penalty. The idea is to start with an initial wavefunction with a well defined number of assets and generate the variational ansatz with a sequence of quantum gates which do conserve spin-excitation number [33].

$$|\Psi_{SWAP,d}(\vec{\theta})\rangle = \prod_{j=2}^{p} \left[\exp\left(-i\sum_i \theta_{ji}\sigma_i^z\right) U_{\sqrt{\text{SWAP}}}\right] \exp\left(-i\sum_i \theta_{1i}\sigma_i^z\right) |\Psi(d)\rangle \,. \tag{21}$$

The initial state $|\Psi(d)\rangle$ has d qubits with value 1, equispaced over the quantum register. Local rotations around the z-axis are interleaved with $U_{\sqrt{\text{SWAP}}}$ gates that implements a 50-50 beam-splitter, partially swapping the qubit excitations

$$U_{\sqrt{\text{SWAP}}} = \prod_{j=1}^{N} \exp\left(i\frac{\pi}{4}\left(\sigma_j^+\sigma_{j+1}^- + \sigma_{j+1}^+\sigma_j^-\right)\right) . \tag{22}$$

4.3. Classical Optimization

As for the classical optimization part, derivative-free optimizers like Constrained Optimization BY Linear Approximation (COBYLA) have proven to be more effective and robust in this task than gradient-based optimizers like gradient descent [34]. Nevertheless, all these methods tend to easily get stuck in local minima when the parameter landscape is highly nonconvex. We checked that this problem is specially acute for the case of low-depth QAOA, in which all the energy structure plus the energy barriers penalizing unfeasible combinations get "compressed" into a few parameters. Thus, in the QAOA scenario, global derivative-free optimizers may help us to improve the success of the minimization. Specifically, we chose the implementation of dual annealing from scipy in Python, which combines a generalization of simulated annealing coupled to a local search. Unlike COBYLA, this method finalizes when the maximum

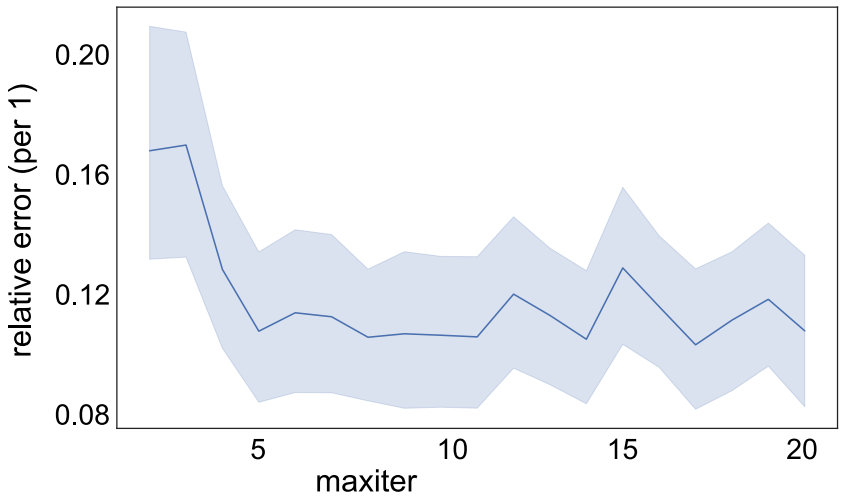

Figure 5. Average relative error Δ achieved by QAOA with one layer and dual annealing (Scipy) as optimizer. Target portfolio $d = 5$.

number of iterations is reached; this is limited by the parameter *maxiter*, which regulates the maximum number of global search iterations.

When using COBYLA, the parameter *tol* marks a convergence criteria; we found that $tol = 0.01$ was enough to ensure convergence and avoid unnecessary iterations at the same time. Yet, the method has to be limited by a total number of iterations, which was set to 2000 because our numerical experiments normally converge within that limit. To attempt a fair comparison with dual annealing, we could try finding a *maxiter* parameter from which the quality of results plateaus. In view of figure 5, that shows the evolution of the relative error for QAOA with one layer and $d = 5$, we conclude that the method rapidly plateaus, being $maxiter = 10$ a reasonable choice. Finally, we have to establish some bounds where the global optimizer is going to be searching: we set the intervals $[0, 2\pi]$ and $[0, \pi]$ for $\vec{\gamma}$ and $\vec{\beta}$ respectively. No further fine tuning was considered apart from the already mentioned.

4.4. Numerical experiments

We have compared numerically the performance of VQE (COBYLA), QAOA (COBYLA), SWAP (COBYLA) and QAOA (dual annealing) using the following procedure. We selected a set of random dates from the whole dataset. For each date we ran each algorithm with a random initial point for every portfolio size (in the case of QAOA the initial point lies within the region previously indicated). Once the classical optimization converged, we measured the quantum state 100 times. Out of the resulting binary numbers, we kept the configuration $\mathbf{x}$ with the lowest energy that satisfies the cardinality constraint $\|\mathbf{x}\|_0 = d$. As a classical reference, we also calculated the best result among 100 repetitions of simulated annealing using the dimod library.

We quantify the performance of the stochastic methods, quantum and classical, using two metrics. Figure 6a compares the relative error obtained with different circuits

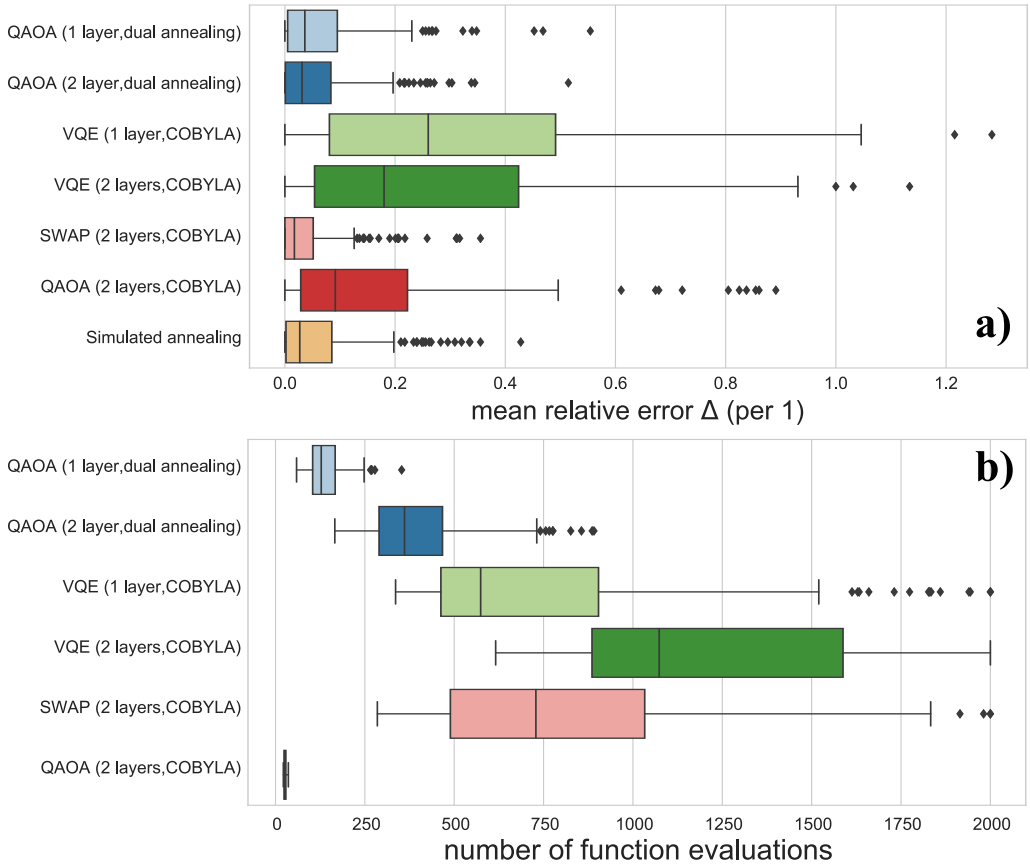

Figure 6. Comparison of QAOA, VQE and SWAP ansätze, sampling from random days,random seeds and ratios d/N, and picking the best result from 100 measurements. a) Box plot of relative errors, dimod's simulated annealing with *reads* = 100 for comparison. b) Box plot of number of function evaluations.

and optimization methods, taking as basis the error of the optimal solution obtained through diagonalization, $\Delta = \frac{P_{\text{err}} - P_{\text{err}}^{\text{opt}}}{P_{\text{err}}^{opt}}$. Figure 6b focuses on the number of evaluations for each quantum ansatz and optimization method. In all cases, we explore portfolio sizes in the interval $[5, 10]$, representing between 1/3 and 2/3 of the universe of equities (reasonable for a multi-step pruning algorithm). Finally, to account for statistical fluctuations, we apply the non-parametric Wilcoxon rank-sum test, which tests the null hypothesis that two sets of measurements are drawn from the same distribution—the alternative hypothesis is that values in one sample are more likely to be larger than the values in the other one.

We can offer some conclusions from this study. First, the SWAP ansatz offers the lowest average relative error ($\langle\Delta\rangle = 4\%$), followed closely by QAOA with dual annealing (6%). This is confirmed by the rank-sum test with a p-value $p = 0.0072$ for a significance level $\alpha = 0.05$. VQE, in comparison, offers poor performance (31%). An additional VQE layer improves the quality, but requires twice the number of function evaluations. We can see that COBYLA quickly gets trapped in local minima for QAOA, leading to worse relative error (16%). Finally, an advantage of QAOA over SWAP stems from the fact

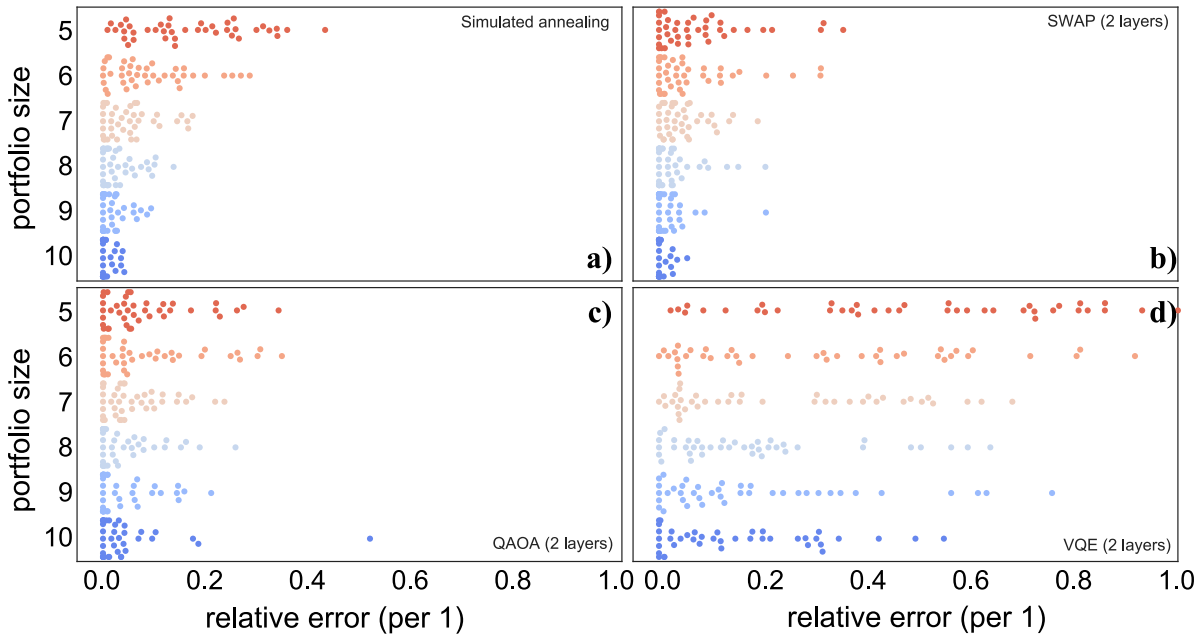

Figure 7. Swarm plot of relative errors Δ for portfolio sizes in the interval $[5, 10]$, using (a) a classical simulated annealing method or a quantum variational method with two layers of (b) SWAP, (c) QAOA or (d) VQE.

that it requires less function evaluations to achieve similar relative errors.

We can also look into how the quality of results vary as we increase the ratio d/N. Figure 7 displays swarm plots for each method whereby we can graphically compare the dispersion of results, their average and the probability of finding the global optimum. All methods are consistent in showing a betterment of all these metrics as the portfolio grows. Interestingly, the probability of finding the optimal portfolio for SWAP and QAOA is better than simulated annealing for the hardest instance ($d = 5$); the rank-sum confirms this shift in the distribution with p-values $p = 8.3\times10^{-5}$ and $p = 2.7\times10^{-3}$ respectively. In comparison, the test does not find statistical difference for portfolios larger than 7.

5. Soft-constraint formulation

5.1. Regularized cost function

The QUBO formulation with hard constraints allows us to control very well the portfolio size, but it creates very rough energy landscapes where classical optimizers such as COBYLA get easily trapped. We can improve the convergence of the optimizer by relaxing our control over the portfolio size, changing the cost function to include Lagrange multipliers instead of quadratic penalties. The new regularization term acts as a *chemical potential* that favors different values of the constraint, depending on the regularization parameter λ

$$P_{\text{err}}^{\text{QUBO}_{\text{soft}}} \sim \mathbf{x}^T Q \mathbf{x} + \lambda ||\mathbf{x}||_0. \tag{23}$$

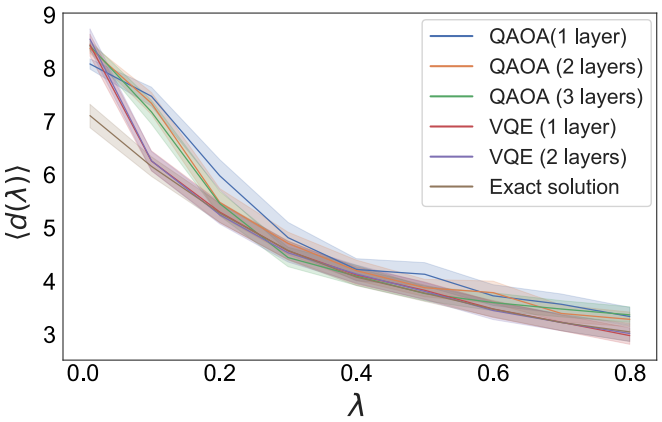

Figure 8. Average portfolio size $\langle d \rangle$ vs. regularization parameter λ. For each value of the regularization parameter, we solve the optimization problem using different anstze and optimizers. We then compute the average portfolio size over a sample of 100 measurements in the final state, and over a random choice of dates of the same benchmark index (OSX).

Financially speaking, λ can also be interpreted as a constant transaction cost for each asset. Although regularization is widely used in machine learning, this particular form is not so common, because it leads to non-differentiable functions. In our case the regularization amounts to introducing a multiple of the identity matrix $\mathbb{1}$

$$\underset{\mathbf{x}}{\operatorname{argmin}}\, P_{\text{err}}^{\text{QUBO}_{\text{soft}}} = \underset{\mathbf{x}}{\operatorname{argmin}} \left[\mathbf{x}^T \mathbf{D}(\Sigma + \lambda \mathbb{1})\mathbf{D}\mathbf{x} - 2\mathbf{x}^T \mathbf{D}\mathbf{g}\right], \tag{24}$$

updating the diagonal elements of Q.

A drawback of this method is that we cannot predict the portfolio size d that is most likely to be selected by any value of the regularization parameter λ. Let us recall, however, that the index tracking optimization is to be repeated multiple times, with minor variations in the data, but not the structure of the problem. Under such circumstances, we have found that indices are stable enough to develop a monotonic relation between the regularization parameter and the average portfolio size. The confidence interval informs us that, for a given λ, the average size remains fairly stable for different random initial point and different dates.

The monoticity of the function $\langle d(\lambda) \rangle$ implies the existence of an optimal λ^* satisfying $\langle d(\lambda^*) \rangle \approx d_{target}$ for each portfolio target. Therefore, to generate portfolios of a target size d, we must tune‖ λ so that the average portfolio produced by our quantum state coincides with d as much as possible. The previous graph indicates that such tuning only needs to happen at the beginning of a series of optimizations over different time windows. Small deviations from the optimal λ^* will not cause a failure to recover the right portfolios—they may affect the quality of the results and the likely of finding low energy configurations with the right size d—, and in any case the value of λ may

‖ Remarkably, the values seem to be independent of the ansatz, perhaps suggesting that we could even train a supervised machine learning model that outputs a suitable λ given a certain matrix Q so as to automatize the parameter tuning.

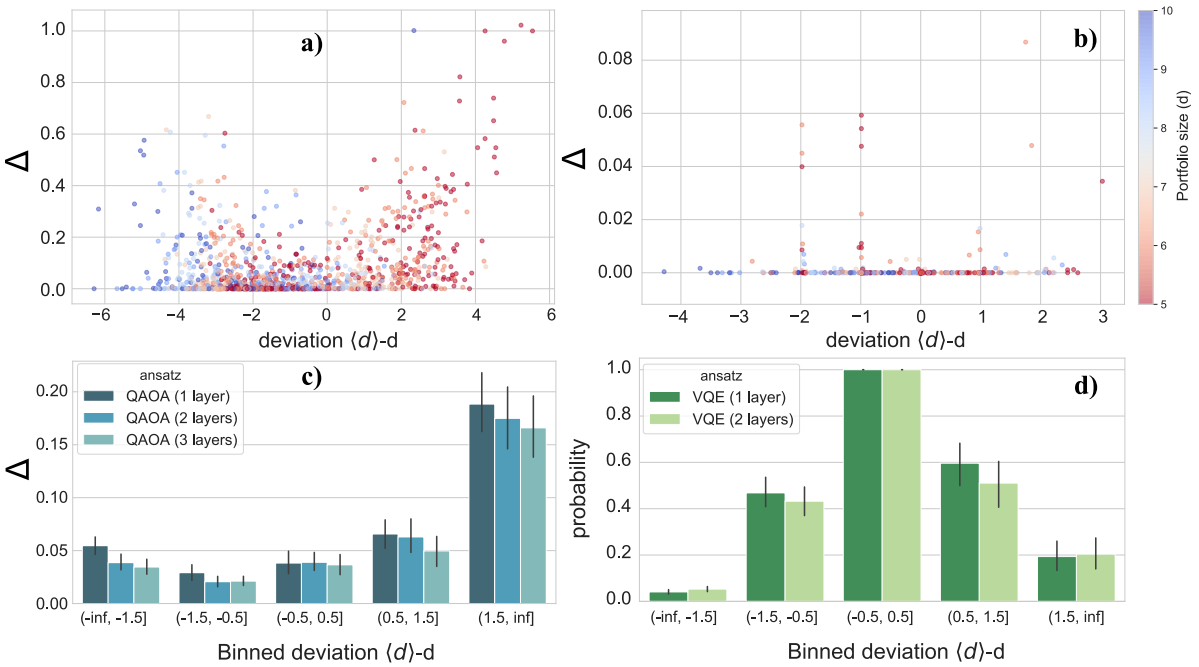

Figure 9. Soft constraint variational optimization performance with QAOA and VQE. Relative error vs. deviation in portfolio size $\langle d \rangle - d$ for (a) QAOA or (b) VQE, both with one layer, solved using and COBYLA. (c) Relative error vs. binned deviation in portfolio size, for QAOA with 1, 2 and 3 entangling layers. (d) Probability of success at sampling a portfolio of size d, for VQE with 1 and 2 entangling layers.

be dynamically adjusted.

5.2. Performance study

To illustrate this effect, in figure 9a we present a scatter plot of relative errors achieved by QAOA with 1 layer and COBYLA as a function of the difference between the mean $\langle d \rangle$ and the desired size d (the baseline here is the hard-constraint exact solution for a fair comparison with previous section); a binned version is shown in plot 9c, also comparing the impact of additional layers. We employed the same numerical procedure described earlier, but in the last step we select the portfolio with the lowest energy satisfying the cardinality constraint from the 100 measurements (if such configuration exists). Effectively, the relative error progressively worsens as we separate from λ^*, yet we can expect good results within two units around λ^*. The asymmetry stems from the fact that it is more likely to find smaller $d's$ on the right hand of the optimal point, which statistically yields greater dispersion as seen in the previous section. We can also see that introducing layers slightly enhance the results; a rank-sum test identifies a conclusive difference between the first and second layer distribution with p-value $p = 1.5 \times 10^{-4}$, and a somewhat less conclusive $p = 0.024$ between the second and third layer.

A strikingly contrasting scenario shows up when repeating the same analysis for VQE, as the scatter plot 9b reveals: sampling the global optimum is very likely happen as long as VQE succeeds in returning a configuration satisfying the cardinality constraint.

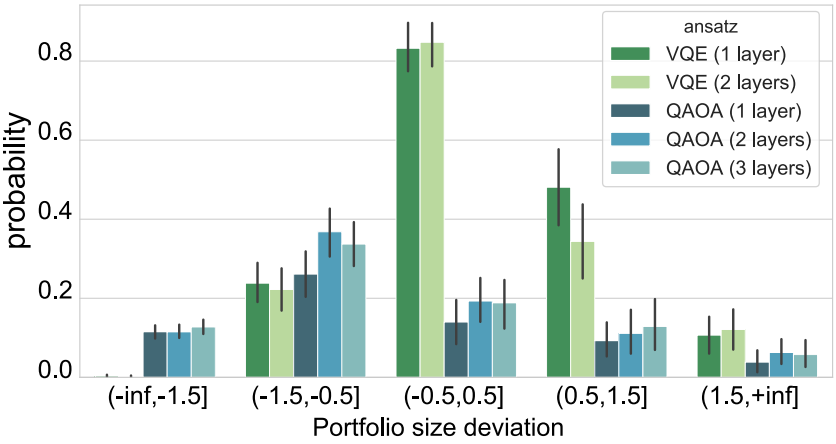

Figure 10. Probability of sampling the global optimum as a function of the binned deviation $\langle d \rangle - d$, comparing various ansätze and number of entangling layers.

Deterioration is precisely noticed in the probability of reaching such feasible portfolio, rapidly decaying as we deviate from the optimal λ^* as plotted in figure 9d (unlike QAOA, which remains fairly stable for greater deviations). In essence, VQE almost guarantees to pinpoint the global optimum provided that we are close enough to the optimal regularization parameter $\lambda \simeq \lambda^*$. Introducing an additional layer does not seem to provide any advantage (the rank-sum tests throws an inconclusive $p = 0.54$).

A metric that somehow summarizes our previous discoveries is the probability of sampling the global optimum, compared in the bar plot 10 for both VQE and QAOA. VQE clearly outstrips QAOA in this regard, but when comparing the number of function evaluations (bar plot 11) we note that it requires around 8 times more calls to the quantum processor. In the light of these results, we conclude that the soft-constraint QUBO encoding improves enormously the energy landscape with respect to the hard-constraint version, leading to better metrics. Provided we are close enough to the optimal regularization parameter λ^*, QAOA improves performance in both relative error and calls even when using COBYLA, where previously we had to rely on a global optimizer. VQE offers high probability of reaching the global optimum but is less robust to large deviations from λ^* than QAOA, and it also demands many more function evaluations.

We have analyzed results for a single pruning step, but the soft-constraint formulation can also be merged into the multi-step algorithm. Since it is not necessary to keep an exquisite control over the portfolios sizes at intermediate steps, a pruning schedule may include a schedule also for the regularization parameters $\lambda_0 \geq \lambda_1 \geq \ldots \geq \lambda_k$. We would start from a high enough value λ_0 to explore larger portfolios, and carefully select the regularizers close to the end, so that $\lambda_k \to \lambda^*$. Another idea is to apply different ansatz and constraint formulations at different stages of the pruning. One could use a VQE-like ansatz with a soft encoding for the intermediate steps, where no fine tuning of λ is needed, and switch to a low-depth QAOA with hard constraints

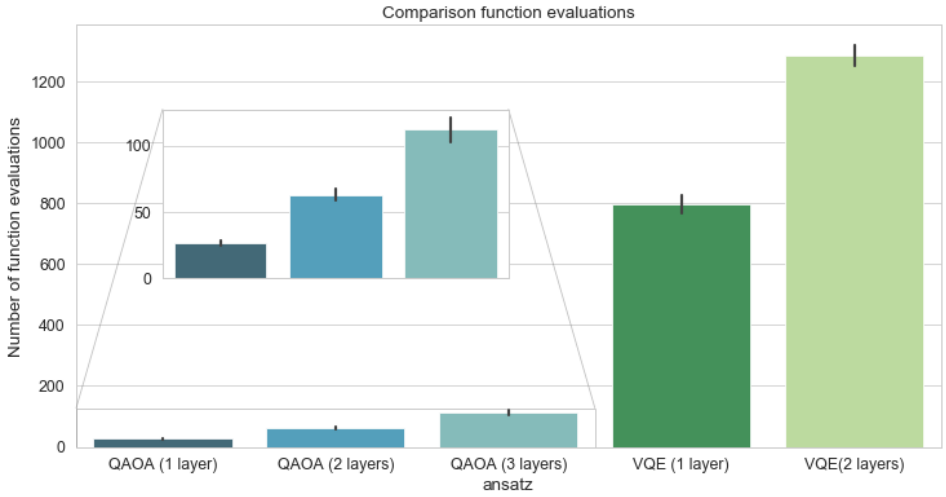

Figure 11. Bar plot of average number of function evaluations, comparison between QAOA and VQE with $|\langle d \rangle - d| < 1$

at the end, to have control over the final portfolio size d.

6. Conclusions

In this work we have introduced an algorithm that iteratively discards investment alternatives to arrive to a basket of assets with similar performance with respect to a financial benchmark. The selection process is driven by quantum optimization algorithms such as QAOA. Unlike other graph-like problems used as a reference in this context (e.g. Max-Cut), index tracking naturally poses a dense quadratic structure, normally associated with hard optimization instances.

Through numerical experiments with real data, we found that the performance of the pruning algorithm depends on the combination of three factors: the quantum ansatz, the classical optimizer, and the mathematical encoding. Figure 12 displays a ranking of all the methods studied regarding averages obtained for three key metrics: the relative error with respect to the exact solution, the number of function evaluations and the probability of sampling the global optimum.

As we can see, the mathematical encoding is the most prominent actor determining the performance for both error and sampling. Introducing penalty terms in a QUBO formulation to account for cardinality constraints drastically increases the hardness of the problem. Soft constraints or quantum circuits designed to search in appropriate sub-spaces become practical alternatives. Local rotations in the VQE ansatz greatly decreases the speed of the algorithm, but it may also lead to high probability of sampling the optimum. QAOA seems well-balanced between speed and accuracy.

In summary, our results point to a promising quantum heuristic for portfolio optimization with cardinality constraints [22], like index tracking as a prototypical example. Experimental tests in real hardware with larger indices remains as next step.

Further constraints ensuring diversification can also be included in our algorithm, which makes it a realistic proposal for the finance industry in the advent of the NISQ era.

Even though we have developed our heuristic pruning algorithm in the context of a particular financial application, it is general enough to be applicable in many other contexts. For instance, it can be applied for selecting variables in a multi-variate linear regression [35] (a common problem in machine learning and econometrics) with a different heuristic with respect to local-search algorithms like threshold accepting [36, 37]. Other applications in machine learning include the sparse Principal Component Analysis [38]; in ensemble machine learning, it also serves as an extension of the QBoost algorithm [39] for selecting K weak learners out of a collection of N predictors. Our results also have applications in other technical disciplines, like compressed sensing/sampling for signal processing [40], or the capacitated facility location problem for supply chain [41].

Acknowledgments

D. Porras and J.J. García Ripoll acknowledge funding from Spanish project PGC2018-094792-B-I00 (MCIU/AEI/FEDER, UE), CSIC Research Platform PTI-001, and CAM/FEDER Project No. S2018/TCS4342 (QUITEMAD-CM). We thank Ecolástico Sanchez for fruitful discussions about the index tracking problem.

References

[1] Paolo Brandimarte. *Numerical Methods in Finance and Economics: A MATLAB-Based Introduction*. Statistics in Practice. Wiley, 2013.

[2] Romn Ors, Samuel Mugel, and Enrique Lizaso. Quantum computing for finance: Overview and prospects. *Reviews in Physics*, 4:100028, 2019.

[3] Gili Rosenberg, Poya Haghnegahdar, Phil Goddard, Peter Carr, Kesheng Wu, and Marcos López De Prado. Solving the Optimal Trading Trajectory Problem Using a Quantum Annealer. *IEEE J. Sel. Top. Signal Process.*, 10(6):1053–1060, sep 2016.

[4] Andrew Milne, Maxwell Rounds, and Phil Goddard. Optimal feature selection in credit scoring and classification using a quantum annealer. *1QB Inf. Technol. (White Paper)*, 2017.

[5] Patrick Rebentrost and Seth Lloyd. Quantum computational finance: quantum algorithm for portfolio optimization, 2018.

[6] Iordanis Kerenidis, Anupam Prakash, and Dániel Szilágyi. Quantum algorithms for portfolio optimization. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. ACM, oct 2019.

[7] Mark Hodson, Brendan Ruck, Hugh Ong, David Garvin, and Stefan Dulman. Portfolio rebalancing experiments using the quantum alternating operator ansatz, 2019.

[8] Panagiotis Kl. Barkoutsos, Giacomo Nannicini, Anton Robert, Ivano Tavernelli, and Stefan Woerner. Improving variational quantum optimization using CVaR. *Quantum*, 4:256, apr 2020.

[9] Patrick Rebentrost, Brajesh Gupt, and Thomas R. Bromley. Quantum computational finance: Monte Carlo pricing of financial derivatives. *Phys. Rev. A*, 98(2):022321, aug 2018.

[10] Stefan Woerner and Daniel J. Egger. Quantum risk analysis. *npj Quantum Inf.*, 5(1):15, dec 2019.

[11] Román Orús, Samuel Mugel, and Enrique Lizaso. Forecasting financial crashes with quantum computing. *Phys. Rev. A*, 99(6):060301, jun 2019.

[12] Xiaojin Zheng, Xiaoling Sun, Duan Li, and Jie Sun. Successive convex approximations to cardinality-constrained convex programs: a piecewise-linear dc approach. *Computational Optimization and Applications*, 59(1-2):379–397, 2014.

[13] Juan Francisco Monge. Cardinality constrained portfolio selection via factor models. *arXiv preprint arXiv:1708.02424*, 2017.

[14] Roel Jansen and Ronald Van Dijk. Optimal benchmark tracking with small portfolios. *The journal of portfolio management*, 28(2):33–39, 2002.

[15] John E Beasley, Nigel Meade, and T-J Chang. An evolutionary heuristic for the index tracking problem. *European Journal of Operational Research*, 148(3):621–643, 2003.

[16] Thomas F Coleman, Yuying Li, and Jay Henniger. Minimizing tracking error while restricting the number of assets. *Journal of Risk*, 8(4):33, 2006.

[17] Rubén Ruiz-Torrubiano and Alberto Suárez. A hybrid optimization approach to index tracking. *Annals of Operations Research*, 166(1):57–71, 2009.

[18] Manfred Gilli and Evis Këllezi. The threshold accepting heuristic for index tracking. In *Financial engineering, e-commerce and supply chain*, pages 1–18. Springer, 2002.

[19] Mohsen Varsei, Naser Shams, Behnam Fahimnia, and Abbas Yazdanpanah. A heuristic approach to the index tracking problem: a case study of the tehran exchange price index. *Asian academy of management Journal*, 18(1):19, 2013.

[20] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[21] QC Ware Corp. A quadratic unconstrained binary optimization problem formulation for single-period index tracking with cardinality constraints, 2018.

[22] Rafael Moral-Escudero, Rubén Ruiz-Torrubiano, and Alberto Suárez. Selection of optimal investment portfolios with cardinality constraints. In *2006 IEEE International Conference on Evolutionary Computation*, pages 2382–2388. IEEE, 2006.

[23] Rubén Ruiz-Torrubiano and Alberto Suárez. Hybrid approaches and dimensionality reduction for portfolio selection with cardinality constraints. *IEEE Comput. Intell. Mag.*, 5(2):92–107, may 2010.

[24] D-Wave Systems Inc. Dimod, a shared api for qubo/ising samplers. Public repository https://github.com/dwavesystems/dimod.

[25] Manfred Gilli and Enrico Schumann. *Portfolio optimization with Threshold Accepting: a practical guide*. Elsevier, 2009.

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

[27] Edward Farhi and Aram W Harrow. Quantum Supremacy through the Quantum Approximate Optimization Algorithm. *Bull. Am. Phys. Soc.*, Volume 62,, 2016.

[28] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New J. Phys.*, 18(2):023023, feb 2016.

[29] Nikolaj Moll, Panagiotis Barkoutsos, Lev S Bishop, Jerry M Chow, Andrew Cross, Daniel J Egger, Stefan Filipp, Andreas Fuhrer, Jay M Gambetta, Marc Ganzhorn, Abhinav Kandala, Antonio Mezzacapo, Peter Mller, Walter Riess, Gian Salis, John Smolin, Ivano Tavernelli, and Kristan Temme. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 3(3):030503, jun 2018.

[30] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin. Quantum approximate optimization algorithm: performance, mechanism, and implementation on near-term devices. *arXiv preprint arXiv:1812.01041*, 2018.

[31] Eleanor G Rieffel, Stuart Hadfield, Tad Hogg, Salvatore Mandrà, Jeffrey Marshall, Gianni Mossi, Bryan OGorman, Eugeniu Plamadeala, Norm M Tubman, Davide Venturelli, et al. From ansätze to z-ates: Nasa view of quantum computing. *Future Trends of HPC in a Disruptive Scenario*, 34:133, 2019.

[32] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. Hardware-efficient variational quantum eigensolver for small

molecules and quantum magnets. *Nature*, 549(7671):242246, Sep 2017.

[33] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor Rieffel, Davide Venturelli, and Rupak Biswas. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. *Algorithms*, 12(2):34, feb 2019.

[34] Giacomo Nannicini. Performance of hybrid quantum-classical variational heuristics for combinatorial optimization. *Physical Review E*, 99(1):013304, 2019.

[35] Alan Miller. *Subset selection in regression*. CRC Press, 2002.

[36] Antanas Zilinskas. Optimization heuristics in econometrics: Applications of threshold accepting, 2003.

[37] Manfred Gilli, Dietmar Maringer, and Enrico Schumann. *Numerical methods and optimization in finance*. Academic Press, 2019.

[38] Alexandre d'Aspremont, Laurent E Ghaoui, Michael I Jordan, and Gert R Lanckriet. A direct formulation for sparse pca using semidefinite programming. In *Advances in neural information processing systems*, pages 41–48, 2005.

[39] Hartmut Neven, Vasil S Denchev, Geordie Rose, and William G Macready. Training a binary classifier with the quantum adiabatic algorithm. *arXiv preprint arXiv:0811.0416*, 2008.

[40] Emmanuel J Candès and Michael B Wakin. An introduction to compressive sampling. *IEEE signal processing magazine*, 25(2):21–30, 2008.

[41] Jesús M Velásquez-Bermúdez, Marzieh Khakifirooz, and Mahdi Fathi. *Large Scale Optimization in Supply Chains and Smart Manufacturing: Theory and Applications*, volume 149. Springer Nature, 2019.

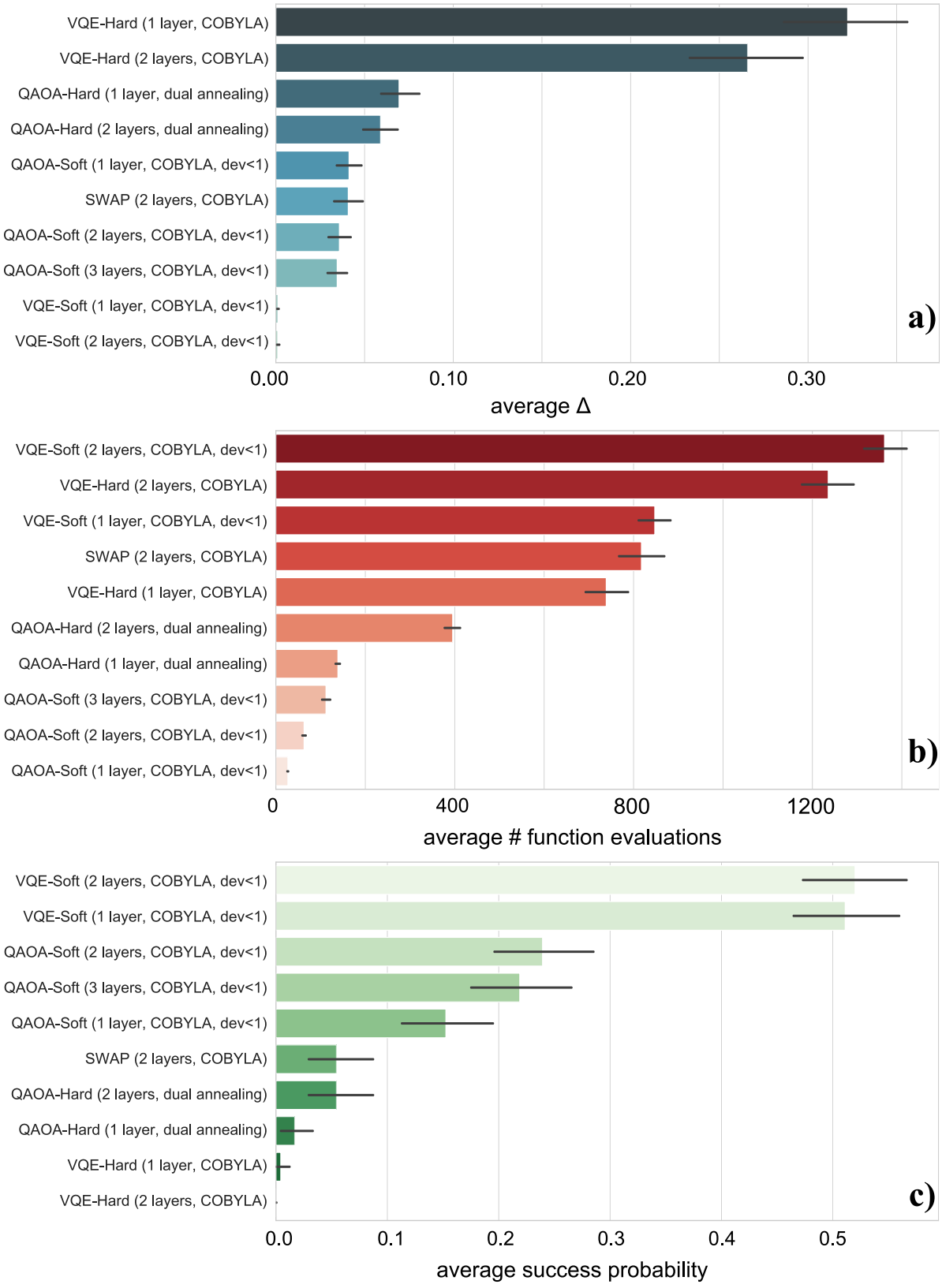

Figure 12. Ranking of methods for (a) average relative error (b) average number of function evaluations and (c) probability of sampling the global optimum. dev<1 stands for $|\langle d\rangle - d| < 1$