White Paper

A quadratic unconstrained binary optimization problem formulation for single-period index tracking with cardinality constraints

QC Ware Corp.[a]

[a]*QC Ware Corp., 125 University Ave, Suite 260, Palo Alto, CA 94301*

Abstract

In this white paper we mathematically formulate a discretized version of the index-tracking problem with sparsity constraints. Such problems have become increasingly important with the rapid rise of index funds and smart beta strategies. Such investing typically requires a risk model to replicate an index with potentially thousands of securities, however, the actual trades may be subject to sparcity, liquidity or lot size constraints. The resulting optimization problem is in the form of a quadratic unconstrained binary optimization (QUBO) problem. One of the target uses for near-term quantum computers—both quantum annealers, and circuit-model quantum computers running the Quantum Approximate Optimization Algorithm—is as heuristic solvers of QUBO problems. By creating a QUBO formulation of the index-tracking problem, we have shown how to formulate the problem in a way that can in principle be run on near-term quantum computers. We verified the correctness of the QUBO problem formulation by solving several problem instances in IBM CPLEX, both using a canonical formulation and using the QUBO formulation. We also characterize the time CPLEX takes to find an (absolute, not approximate) optimal solution for a variety of instances, which provides an upper bound for the time needed for a classical solver to solve these instances.

Keywords: Portfolio optimization, Sparse index tracking, QUBO, Quantum annealing, CPLEX

1. Introduction

Index tracking and smart beta, which typically involves replicating a dynamic or non-market capitalization weighted index, are a significant part of the equity index market and are rapidly growing in the fixed income and commodity spaces. When an index has relatively few, highly liquid securities it is possible to just purchase all the index securities in the appropriate proportion. However, and this is particularly true in fixed income, if the index has many securities (such as thousands), most of which do not trade on a given day, and may only be tradable in round lots, and/or do not trade on an electronic exchange, it is necessary to do some optimization. This is usually done with a risk model, often assumed to be a linear model, to minimize the expected volatility of the portfolio relative to its index. Unfortunately, adding sparsity and round lot constraints can make the optimization computationally difficult.

Index tracking, an important operational research problem in quantitative finance, consists of reproducing the performance of a market index by investing in a subset of the assets within the index. The index tracking problem can arise frequently for fund managers, who follow a strict set of constraints to guarantee a minimum level of return. Often, following these constraints entails matching the performance of a large financial index, without incurring significant transaction costs from activities such as portfolio rebalancing. We emphasize the relevance of fund management in modern finance, noting that in 1998, over 100 billion U.S. dollars were invested in passive funds in the U.S. alone (Sorenson et al., 1998), and that by 2015, this number climbed to over two trillion U.S. dollars.

Two main approaches to index tracking are physical replication and synthetic replication. Physical replication, which is much more common (see for example Naumenko and Chystiakova (2015); Amenc et al.

Email address: `info@qcware.com` (QC Ware Corp.)

(2012)), involves investing directly in assets contained in the index. Synthetic replication uses derivative contracts, such as options, futures, and swaps, to track the index. In physical replication, full replication of the index—composing the portfolio of all assets in the index—maximizes tracking accuracy in a frictionless market since the tracking portfolio and the index will have the same normalized shape (Strub and Baumann, 2018). However, full replication is suboptimal in a market with frictions, e.g. due to transaction costs involved in rebalancing the tracking portfolio (Connor and Leland, 1995; Strub and Baumann, 2018). Moreover, full replication is often not feasible due to liquidity constraints or the size of the index's asset universe. Thus, a cardinality or sparsity constraint is often enforced when formulating an index tracking problem (Ruiz-Torrubiano and Suárez, 2009; Beasley et al., 2003).

From a mathematical perspective, there are modeling choices to make when formulating and solving index tracking problems. A common choice is to phrase the problem variationally, where the optimal asset allocation or portfolio construction is the minimum of some objective function subject to constraints, such as a cardinality constraint as suggested above. We note that additions like cardinality constraints can introduce combinatorial complexity into the problem; minimizing tracking error subject to cardinality constraints was shown to be NP-hard by Coleman et al. (2006). Thus, even within the realm of variational formulations, there are many ways to ultimately formulate and solve an index tracking optimization problem, and computational tractability and efficiency plays a significant role in model design and selection.

Recent developments in quantum computing hardware has led to a renewed interest in the capabilities of these machines for potentially solving certain optimization problems faster than is possible with any classical computer. Each of the quantum computing prototypes that have been built so far can be broadly classified as being either a *universal circuit-model quantum computer*, or a *quantum annealer*. Quantum annealers offer a restricted form of quantum computation—they are special-purpose optimization machines—whereas circuit-model quantum computers can run quantum algorithms that extend beyond optimization. At the present time however, the devices with the most number of physical quantum bits, or *qubits*, are the quantum annealers, with an order of magnitude more qubits than the largest circuit-model quantum computer. Quantum annealers are based on the principle of adiabatic quantum computation, and are typically designed to try find the ground state of a given Ising optimization problem. A class of optimization problems called quadratic unconstrained binary optimization (QUBO) problems belong to the class of NP-hard problems, and are easily transformed into the Ising Hamiltonian form, and hence are also amenable to solution on quantum annealers. More generally, any optimization problem that can be cast as a QUBO problem is in principle solvable on a quantum annealer.

This work presents a QUBO formulation of a single-period index-tracking problem. We begin by first formulating a constrained optimization problem for index tracking, and introduce discretization into it. Then through a sequence of mathematical transformations, the problem is transformed into QUBO form which can then be solved using a quantum annealer. In fact, two different QUBO formulations are presented, each of which encodes the problem using different number of variables, the first based on a *unary encoding* scheme, and the second based on a *binary encoding* scheme for the problem. The proposed QUBO formulations are tested for correctness using CPLEX. The dataset that we use for the study is a snapshot of all assets and their relative makeup of the Barclay's US High Yield 2% Index of US High Yield Bonds.

2. Related Work

Significant research has been conducted on index tracking. In rigorously defining the index tracking problem, a number of metrics for tracking error which index tracking seeks to minimize, have been proposed in the literature. As stated by Ruiz-Torrubiano and Suárez (2009), tracking error is commonly defined in terms of the correlation between an index's returns and that of the tracking portfolio, or in terms of the variance of the difference of the returns of the index and the returns of the tracking portfolio (Markowitz, 1989; Buckley and Korn, 1998; Shapcott, 1992; Roll, 1992). It has been noted that errors based on the variance of the difference in returns fail to penalize tracking portfolio return profiles that are constant offsets of index return profiles (Beasley et al., 2003; Ruiz-Torrubiano and Suárez, 2009); however, this situation is not encountered in practice, and variance formulations remain popular definitions of tracking error. Alternatively, it has been proposed to use the mean squared difference in the returns as the tracking error metric (Ruiz-Torrubiano and Suárez, 2009; Beasley et al., 2003; Lobo et al., 2007), which avoids the constant-offset edge case. One can also use absolute deviations of the returns of the tracking portfolio and index in defining tracking error

(Clarke et al., 1994; Wang et al., 2012; Rudolf et al., 1999), an approach that has the advantage of being linearizable. We employ a variance based definition of tracking error in this work.

A number of works have also considered constrained versions of the index tracking problem, which more realistically emulate the challenge faced by a passive fund manager in replicating an index. Jansen and Van Dijk (2002), and Coleman et al. (2006) both enforce a limited number of assets in their tracking portfolio. They minimize a weighted combination of the continuous tracking error and the discrete number of assets in the tracking portfolio, with the latter being approximated continuously. Another common constraint requires that the total value of the tracking portfolio plus any transaction costs does not exceed the total capital available to construct the portfolio (Strub and Baumann, 2018; Guastaroba and Speranza, 2012). In Wang et al. (2012), a conditional value at risk constraint is introduced in order to limit the downside risk of the tracking portfolio. For additional surveys of the index tracking literature, the reader is referred to Beasley et al. (2003), Canakgoz and Beasley (2009), and Karlow (2012).

Given the focus of the present work, we also remark upon the literature pertaining to binary optimization, especially QUBO problems. Such problems often arise in the context of combinatorial search problems, including NP-complete and NP-hard problems. Binary optimization problems have been studied and applied across an array of different fields; an early example is Hammer and Shlifer (1971), which describes applying them to the problem of selecting locations for service stations. Financial applications of binary optimization date back to around 1970 with works like Laughhunn (1970). We note that constrained binary optimization problems can often be reformulated as unconstrained problems (Hansen, 1979), and thus both constrained and unconstrained binary optimization problems may often be treated similarly.

A number of exact and approximate solution methods for QUBO problems have been developed. Lucas (2014) described Ising formulations of all of Karp's 21 NP-complete problems (Karp, 1972), which are equivalent to QUBO problems as remarked earlier. Indeed, the exact solution of QUBO problems is known to be NP-hard in general (Pardalos and Jha, 1992). Thus, exact solvers, commonly based on branch-and-bound techniques, are practically limited to solving problems for up to a few hundred variables (Kochenberger et al., 2014). Promising developments continue to be made with exact solvers, both in the research community as well as in the commercial sphere, where solvers like CPLEX have been developed that are successful on moderately-sized QUBO problems (Billionnet and Elloumi, 2007). In contrast to exact methods, heuristic and metaheuristic approaches greatly increase the size of the problem that can be solved, such as set partitioning problems with $15,000$ variables (Lewis et al., 2008) and set packing problems with $2,000$ variables (Alidaee et al., 2008), though obviously with the price of inexact solutions. Of particular note is the simulated annealing heuristic for QUBO problems, studied by Katayama and Narihisa (2001). For a detailed survey on the QUBO problem, including various solution techniques and applications, the reader is referred to Kochenberger et al. (2014).

In a QUBO problem, the objective function is a quadratic, with each variable being either 0 or 1. In contrast, in a Ising Hamiltonian problem, each variable is either 1 or -1. Consequently, the transformation of a QUBO problem to a Ising Hamiltonian problem is easily achieved with a simple variable transformation $x \mapsto 2x - 1$. Thus an efficient algorithm to find the minimum energy state of the Ising Hamiltonian, which is also called the *Ising problem*, would imply an efficient solution to the QUBO problem. The idea that NP-complete satisfiability problems could be solved by minimizing Ising Hamiltonians, using the principle of adiabatic quantum computation or *quantum annealing* (Kadowaki and Nishimori, 1998; Santoro et al., 2002; Johnson et al., 2011), was first introduced by Farhi et al. (2000), and later analyzed and developed further in Farhi et al. (2001), and Childs et al. (2001). Collectively, these papers constitute much of the foundation of today, for the widely held belief that certain instances of QUBO problems could be solved much faster using quantum annealing than any other classical algorithm running on a classical computer.

Since its introduction, a lot of effort has gone into finding practical applications of quantum annealing. The literature is too vast to present a complete survey here, and we mention only a few of the notable works. Early applications of adiabatic quantum computation include finding cliques in graphs (Childs et al., 2000), optimization on regular graphs (Farhi et al., 2012), and clustering (Kurihara et al., 2009). In the field of machine learning, some interesting applications have been found very recently such as sampling from restricted Boltzmann machines (Adachi and Henderson, 2015; Biamonte et al., 2017), reinforcement learning for finite episodic games (Neukart et al., 2018), free energy based reinforcement learning (Levit et al., 2017), and Dirichlet process mixture models (Sato et al., 2013). Applications of quantum annealing have also been found for traffic flow optimization (Neukart et al., 2017), and most notably in the field of finance for solving the optimal trading trajectory problem (Rosenberg et al., 2016). All these applications can be grouped into two categories—either solving the problem of finding the minimum energy state of an Ising Hamiltonian, or sampling a distribution using a quantum annealer. The work that we present in this paper falls in the first category.

It should be noted that it was proved by Aharonov et al. (2008) that quantum annealing is equivalent to universal quantum computing in the circuit model for noise free systems; but this is not true in any realistic quantum system with noise. While efficient error correction mechanisms exist for the circuit model of quantum computation (Shor, 1996; Preskill, 1998; Gottesman, 1998), the same is not known to be true in general for quantum annealing, and this is a subject of active research (Jordan et al., 2006; Pudenz et al., 2014, 2015). Because of the equivalence result however, it is possible to implement quantum annealing in the circuit model of quantum computation, which means in particular that one can solve the Ising problem also using a universal quantum computer. In fact, such an algorithm already exists, called the quantum approximate optimization algorithm (QAOA) (Farhi et al., 2014), which allows the mapping of the Ising problem to one that is solvable on a circuit model quantum computer.

3. Problem Formulation and Discretization

In this section we introduce the mathematical formulation of the single period index tracking problem that we study in this paper. The term *single-period* means that we only wish to track the index at a fixed instant of time, as opposed to *multi-period* index tracking that attempts to track the index over multiple periods of time. This is typically called a risk model and in most cases a linear risk model like the one we present is sufficient. These simplifications allow us to obtain a simplified problem that captures the overall essence of the index tracking problem, while at the same time avoids the complications of the most general case, and is motivated by the main goal of this white paper which is to study the numerical performance of CPLEX while solving instances of the problem. The choice of simplicity is also motivated a great deal by the constraint of problem size that we can solve presently, in terms of the number of variables, and in a reasonable amount of time.

The metric that we use to evaluate how well the tracking portfolio replicates the index, at some instant of time, is the tracking error variance v. If r_I is the return of the index and r_P is the return of the tracking portfolio, then the tracking error variance is defined as

$$v = \mathrm{var}(r_I - r_P). \tag{1}$$

In our problem formulation, we will assume a linear factor model that decomposes the return of each asset in the index into a linear combination of the underlying market factor returns. Let us assume that we have an index with n assets, and we have a l-degree factor model for it, meaning that there are l factors. Let $w \in \mathbb{R}^n$ be a vector with w_i corresponding to the fractional weight of asset i within the index, $p \in \mathbb{R}^n$ be a vector with p_i representing the fractional weight of asset i in the tracking portfolio, $f \in \mathbb{R}^l$ be a vector with f_j corresponding to the return of factor j, and $L \in \mathbb{R}^{n \times l}$ be a matrix such that L_{ij} represents the coefficient of the return of factor j in the linear decomposition of the return of asset i. Then the factor model representations of the returns of the index and the tracking portfolio are given as

$$r_I = w^T L f, \quad r_P = p^T L f. \tag{2}$$

It should also be noted that the fractional weights of the assets in the index and the tracking portfolio satisfy $\sum_{i=1}^n w_i = \sum_{i=1}^n p_i = 1$. The tracking error variance can then be expressed as

$$v = \mathrm{var}\left((w-p)^T L f\right) = (w-p)^T \mathrm{cov}(Lf)(w-p) = (w-p)^T L \Sigma L^T (w-p), \tag{3}$$

where $\Sigma \in \mathbb{R}^{n \times n}$ is the covariance matrix of the factors defined as $\Sigma_{ij} = \mathbb{E}[(f_i - \mathbb{E}(f_i))(f_j - \mathbb{E}(f_j))]$. As covariance matrices are symmetric positive definite, it follows that $L \Sigma L^T$ is also symmetric and positive definite. A typical goal of index tracking is to choose p such that v is minimized, given all the other quantities appearing in (3), which is the case when no other constraints are present. This problem can be solved efficiently in polynomial time using different classical algorithms, for example interior point methods, and is not of interest to us. A slightly more interesting case happens when we consider a version of the problem with added constraints which we now state. The first constraint we consider imposes a restriction on the number of different assets that the tracking portfolio can hold, which is called a *sparsity constraint*. Specifically we impose that the number of assets in the portfolio cannot exceed an integer $d \in \mathbb{N}$, $1 \leq d \leq n$. In applications d is much smaller compared to n, that is $d << n$, and the sparsity constraint alone makes the resulting problem NP-complete. The second constraint, that we impose for the purposes of keeping the problem simple, ensures that the portfolio has no short selling of assets, that is $p_i \geq 0$ for all $1 \leq i \leq n$. Our goals so far can then be expressed in the form of an optimization problem as follows:

$$\begin{aligned} \underset{p}{\text{minimize}} \quad & (w-p)^T L\Sigma L^T (w-p) \\ \text{subject to} \quad & \sum_{i=1}^{n} p_i = 1 \\ & \sum_{i=1}^{n} \mathbb{1}\{p_i > 0\} \leq d. \end{aligned} \tag{4}$$

3.1. Discretization of the weights of the assets in the tracking portfolio

The optimization problem in (4) is not a suitable candidate for quantum annealing, as the portfolio weights span the continuous range $[0,1]$. To obtain a meaningful problem that can be solved using adiabatic quantum computation, we introduce a third constraint into the problem, namely that each p_i can only take values in a finite set instead of $[0,1]$. In fact in many applications in finance, such a scenario actually comes up when one can only buy and sell assets in integral multiples of a fixed lot size for each asset. In this paper, we will assume for simplicity that the lot sizes are the same for the different assets in terms of dollar value, and thus each p_i takes values over the same finite set, which we take to be uniformly spaced over the interval $[0,1]$. Specifically for a positive integer m', we first let $m = 2^{m'}$, and then choose m uniformly spaced points in the interval $[0,1]$ such that the k^{th} point denoted as z_k is given by $z_k = \frac{k}{m-1}$, for all $0 \leq k \leq m-1$. We next introduce nm binary decision variables $\hat{p}_{ik}$, defined as

$$\hat{p}_{ik} = \begin{cases} 1 & \text{if } p_i = z_k \\ 0 & \text{if } p_i \neq z_k, \end{cases} \tag{5}$$

for all $1 \leq i \leq n$, $0 \leq k \leq m-1$, and impose that $\sum_{k=0}^{m-1} \hat{p}_{ik} = 1$ for each i, which ensures that the portfolio weights corresponding to each of the assets take one and only one value from the discrete set. For convenience we will stack the discrete set of weights z_k into a vector $z \in \mathbb{R}^m$, ordered by the index k, and the binary variables $\hat{p}_{ik}$ into a vector $\hat{p} \in \{0,1\}^{nm}$, ordered by the indices i followed by k. Substituting the relations (5) in the objective function of (4), dropping constant terms, and aggregating all the constraints in one place allows us to write the resulting optimization problem that we want to solve in this paper as follows

$$\begin{aligned} \underset{\hat{p}}{\text{minimize}} \quad & \hat{p}^T Q_1 \hat{p} \\ \text{subject to} \quad & \mathbf{1}^T A_1 \hat{p} = 1 \\ & \sum_{k=0}^{m-1} \hat{p}_{ik} = 1, \quad \forall\, 1 \leq i \leq n \\ & \sum_{i=1}^{n} \hat{p}_{i0} \geq n-d, \end{aligned} \tag{6}$$

where $A_1 = I_n \otimes z^T$ with I_n denoting the $n \times n$ identity matrix, $\mathbf{1} \in \mathbb{R}^n$ is a vector of all ones, and $Q_1 = A_1^T L\Sigma L^T A_1 - 2\,\text{diag}(A_1^T L\Sigma L^T w)$. The optimization problem in (6) uses m binary variables for each asset, for a total of nm variables, in order to be able to represent m discrete uniformly sampled values in the interval $[0,1]$ for the asset's portfolio weight, and hence it will also be referred to as the *unary encoding* problem.

It is possible to come up with an exactly equivalent problem as (6) that reduces the number of variables involved in the optimization problem, that we discuss next. Let us introduce nm' binary decision variables

$\hat{b}_{il} \in \{0,1\}$, for all $1 \le i \le n$, $0 \le l \le m'-1$, with the idea behind these variables being that for each asset i, if for some k we have $\hat{p}_{ik} = 1$, then k admits a unique m'-bit binary (base 2) representation given by $(\hat{b}_{i(m'-1)} \dots \hat{b}_{i0})_2$. The resulting relationships between the decision variables $\hat{p}_{ik}$ and $\hat{b}_{il}$, for all $1 \le i \le n$, can then be easily expressed as

$$\begin{aligned}
\hat{b}_{i0} &= \left(\sum_{k=0}^{m-1} k \mathbb{1}\{\hat{p}_{ik} = 1\} \right) \pmod 2, \\
\hat{b}_{il} &= \left(2^{-l} \sum_{k=0}^{m-1} k \mathbb{1}\{\hat{p}_{ik} = 1\} - \sum_{l'=0}^{l-1} \hat{b}_{il'} 2^{l'-l} \right) \pmod 2, \quad \forall\, 1 \le l \le m'-1, \\
\hat{p}_{ik} &= \begin{cases} 1, & \text{if } k = \sum_{l=0}^{m'-1} \hat{b}_{il} 2^l \\ 0, & \text{if } k \neq \sum_{l=0}^{m'-1} \hat{b}_{il} 2^l \end{cases}, \quad \forall\, 0 \le k \le m-1.
\end{aligned} \tag{7}$$

We will stack the variables $\hat{b}_{il}$ into a vector $\hat{b} \in \{0,1\}^{nm'}$, ordered by the indices i followed by l, and also introduce another vector $\hat{u} \in \{0,1\}^n$ defined as $\hat{u}_i = \mathbb{1}\{p_i > 0\}$, which will serve as indicator variables for the portfolio weights of the corresponding assets being non-zero. The optimization problem (6) is then easily shown to be equivalent to the following problem

$$\begin{aligned}
\underset{\hat{b},\hat{u}}{\text{minimize}} \quad & \hat{b}^T Q_2 \hat{b} \\
\text{subject to} \quad & \mathbf{1}^T A_2 \hat{b} = 1 \\
& \sum_{i=1}^{n} \hat{u}_i \le d \\
& \sum_{l=0}^{m'-1} \hat{b}_{il} \le m' \hat{u}_i, \quad \forall\, 1 \le i \le n \\
& \hat{u}_i \le \sum_{l=0}^{m'-1} \hat{b}_{il}, \qquad \forall\, 1 \le i \le n,
\end{aligned} \tag{8}$$

where $A_2 = I_n \otimes [2^0 \quad \dots \quad 2^{m'-1}]/(2^{m'}-1)$, and $Q_2 = A_2^T L \Sigma L^T A_2 - 2 \operatorname{diag}(A_2^T L \Sigma L^T w)$, with all other quantities defined as before. It is clear that if we consider the number of binary variables, then (8) improves on (6) by using only a total of $n(m'+1)$ variables, as compared to nm variables in the unary encoding case. As $m = 2^{m'}$, this means that we obtain an exponential saving in the number of variables with respect to m in the problem formulation (8), which we will refer to as the *binary encoding* problem. However on the down side, we end up having $2n+1$ inequality constraints in order to be able to implement the third constraint in (6), which is needed for the sparsity of the portfolio, but as we show in the next section, this does not significantly increase the final variable count when the inequality constraints are changed to equality constraints with the introduction of slack variables.

4. Transforming the discrete optimization problems to QUBO form

Our next goal is to transform the binary optimization problems in (6) and (8) to QUBO form that can then be solved using a quantum annealer. The final optimization problems thus obtained are unconstrained optimization problems. The critical aspect of such a transformation is that the set of global minimum of the original problem should remain unchanged after the transformation, i.e. any feasible solution that was a global minimum of the original problem should still remain a global minimum of the transformed problem, and any other solution, either feasible or infeasible, should not become a global minimum of the new problem. We achieve this by first introducing slack variables to convert the inequality constraints to equality constraints following Nocedal and Wright (2006), and then by adding the equality constraints to the objective function as squared penalties, as outlined in Hansen (1979). We briefly describe these steps for the unary and binary problems next.

4.1. Transformation of the unary encoding problem

We first notice that the quantities appearing on the left and right hand sides of the inequality constraint of the discrete optimization problem in (6) are integers, and so we can introduce binary slack variables to account for the difference between the left and right hand sides of this constraint. It should be noted that here again we have a choice of either choosing an unary versus a binary encoding scheme to account for the slack. To be consistent, we will follow a unary encoding strategy for the optimization problem in (6), while in the next section a binary encoding strategy will be adopted for performing similar transformations for the optimization problem in (8).

Let us introduce a vector $\hat{\mu} \in \{0,1\}^d$, and so we have $0 \leq \sum_{i=1}^{d} \hat{\mu}_i \leq d$, which implies that $n - d \leq n - d + \sum_{i=1}^{d} \hat{\mu}_i \leq n$. Thus if we set $\sum_{i=1}^{n} \hat{p}_{i0} = n - d + \sum_{i=1}^{d} \hat{\mu}_i$, then this automatically implies that $n - d \leq \sum_{i=1}^{n} \hat{p}_{i0} \leq n$, which in turn implies the inequality constraint in (6). Thus the optimization problem

$$\begin{aligned} \underset{\hat{p},\hat{\mu}}{\text{minimize}} \quad & \hat{p}^T Q_1 \hat{p} \\ \text{subject to} \quad & \mathbf{1}^T A_1 \hat{p} = 1 \\ & \sum_{k=0}^{m-1} \hat{p}_{ik} = 1, \;\; \forall\, 1 \leq i \leq n \\ & \sum_{i=1}^{n} \hat{p}_{i0} = n - d + \sum_{i=1}^{d} \hat{\mu}_i, \end{aligned} \tag{9}$$

is exactly equivalent to (6). Transformation to QUBO form is subsequently achieved by adding all the equality constraints in (9) as squared penalties to the objective function with a sufficiently large penalty parameter α_1 (Hansen, 1979), leading to the following equivalent unconstrained binary optimization problem, that will also be referred to as the *unary encoding QUBO* problem,

$$\underset{\hat{p},\hat{\mu}}{\text{minimize}} \quad \hat{p}^T Q_1 \hat{p} \;+\; \alpha_1 \left[\left(\mathbf{1}^T A_1 \hat{p} - 1\right)^2 + \sum_{i=1}^{n} \left(\sum_{k=0}^{m-1} \hat{p}_{ik} - 1 \right)^2 + \left(\sum_{i=1}^{n} \hat{p}_{i0} - n + d - \sum_{i=1}^{d} \hat{\mu}_i \right)^2 \right]. \tag{10}$$

4.2. Transformation of the binary encoding problem

The transformation of the binary encoding problem (8) to QUBO form follows an exactly analogous strategy. We introduce slack variables to account for the slack in the inequality constraints of (8), but this time we follow a binary encoding strategy which prevents the introduction of an excessively large number of slack variables into the problem. We first define two positive integers $m_1 = \lceil \log_2(1+d) \rceil$, and $m_2 = \lceil \log_2(m') \rceil$. Then introducing the vectors $\hat{\nu} \in \{0,1\}^{m_1}$, $\hat{\eta} \in \{0,1\}^{m_2}$, and $\hat{\zeta} \in \{0,1\}^{m_2}$, and following a process exactly similar to the unary case, we obtain an optimization problem equivalent to (8) as follows

$$\begin{aligned} \underset{\hat{b},\hat{u},\hat{\nu},\hat{\eta},\hat{\zeta}}{\text{minimize}} \quad & \hat{b}^T Q_2 \hat{b} \\ \text{subject to} \quad & \mathbf{1}^T A_2 \hat{b} = 1 \\ & \sum_{i=1}^{n} \hat{u}_i + \sum_{i=0}^{m_1-1} 2^i \hat{\nu}_i = d \\ & \sum_{l=0}^{m'-1} \hat{b}_{il} + \sum_{l=0}^{m_2-1} 2^l \hat{\eta}_{il} = m' \hat{u}_i, \;\; \forall\, 1 \leq i \leq n \\ & \hat{u}_i + \sum_{l=0}^{m_2-1} 2^l \hat{\zeta}_{il} = \sum_{l=0}^{m'-1} \hat{b}_{il}, \qquad \forall\, 1 \leq i \leq n. \end{aligned} \tag{11}$$

Transformation to QUBO form then follows by choosing a parameter α_2 sufficiently large, and squaring and adding the equality constraints as penalties, leading to the following equivalent problem

$$\begin{aligned}\underset{\hat{b},\hat{u},\hat{\nu},\hat{\eta},\hat{\zeta}}{\text{minimize}} \quad & \hat{b}^T Q_2 \hat{b} \;+\; \alpha_2 \left[\left(\mathbf{1}^T A_2 \hat{b} - 1\right)^2 + \left(\sum_{i=1}^{n} \hat{u}_i + \sum_{i=0}^{m_1-1} 2^i \hat{\nu}_i - d \right)^2 + \sum_{i=1}^{n} \left(\sum_{l=0}^{m'-1} \hat{b}_{il} + \sum_{l=0}^{m_2-1} 2^l \hat{\eta}_{il} - m'\hat{u}_i \right)^2 \right. \\ & \left. + \sum_{i=1}^{n} \left(\hat{u}_i + \sum_{l=0}^{m_2-1} 2^l \hat{\zeta}_{il} - \sum_{l=0}^{m'-1} \hat{b}_{il} \right)^2 \right], \end{aligned} \tag{12}$$

which we will also refer to as the *binary encoding QUBO* problem.

In summary, we have reformulated the unary encoding and binary encoding problems in (6) and (8) as QUBO problems in (10) and (12) respectively. Moreover as we have previously shown, (6) and (8) are equivalent to one another, and thus the QUBO problems (10) and (12) are also equivalent. However if we just count the number of the binary variables involved in each QUBO problem, we find that the binary encoding QUBO only uses $n(\log_2 m + 1) + \lceil \log_2(1+d) \rceil + 2\lceil \log_2 \log_2 m \rceil$ binary variables as compared to the unary encoding QUBO which uses $nm + d$ binary variables, and thus (12) achieves exponential savings in the number of variables used as a function of m, as compared to (10).

5. Numerical Experiments

In this section we perform some numerical experiments to study the relative hardness of the QUBO problems (10) and (12), that we formulated in the last section. We study how the time to solution scales for the unary and binary encoding QUBO problems when we vary different parameters such as the number of assets in the portfolio n, the parameter m' that controls the level of discretization of the interval $[0, 1]$, and the sparsity parameter d. All the results presented in this section were performed using the CPLEX optimization library.

5.1. Generating the data set for the numerical study

The data set that we use for our numerical study is a real data set that was provided by the Western Asset Management Company, that comprises of a snapshot of the relative makeup of the Barclay's US High Yield 2 % index of US High Yield Bonds. The entire data set comprises of 2086 assets, while the factor model representation contains a total of 57 factors. Both the covariance matrix for the factors Σ, as well as the matrix of factor weights L were available for this data set.

The first step in the data set generation process for our experiments involves choosing subsets of the provided data set. For each value of the number of assets n that shows up in any of our numerical experiments, we created a set of 10 different data sets each containing n assets, each of which was created by randomly sampling from the original data set with 2086 assets. Each such random sampling process entails choosing n assets randomly out of 2086, which then completely determines the corresponding factor model L for those selected assets. We form the matrix $L\Sigma L^T$ for each such selection which is then stored. For each n, the 10 created data sets were used in all subsequent experiments, without any change.

5.2. Verifying the correctness of the QUBO formulations

The correctness of the QUBO problems (10) and (12) were verified by solving these problems using CPLEX, and then comparing the obtained solutions against their constrained counterparts (6) and (8), which were also solved using CPLEX. It should be noted that in all these experiments using CPLEX, we solve for the absolute global minimum of the objective function. In fact, since all these four problems are equivalent, it means that we should expect to get the same global minimum for identical choices of parameters for the problems in terms of n, m', and d.

To this extent, we performed this test for different levels of coarseness of discretization by varying the parameter $m' = 1, 2, 3$, different number of assets $n = 1, \ldots, 10$, and different cardinality values $d = 1, \ldots, \lfloor n/2 \rfloor$. Moreover, for each triplet of parameters (n, m', d), we performed this test for each of the 10 data sets that we created for that value of n. We got identical results on all these cases tested for all the four optimization problems. These tests provide a numerical proof of verification that the QUBO formulations of the unary and binary encoding problems are correct. While these numerical tests were not strictly necessary to prove their correctness, which in fact follows simply by mathematical reasoning as has been done in the previous sections, the numerical tests were done to rule out any programming errors.

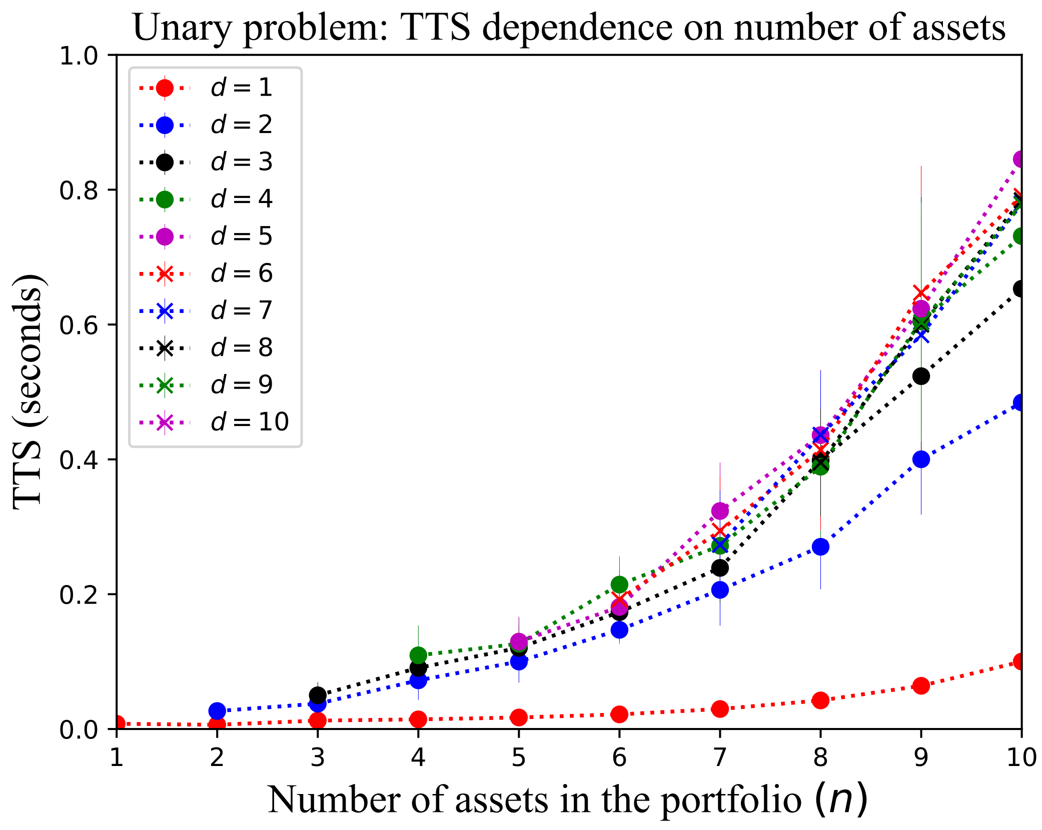

Figure 1: Variation of the time to solution (TTS) with respect to the number of assets n in the portfolio, for the unary encoding QUBO problem (10), where we have plotted the average TTS along with their standard deviation for each value of d and n. Each of the curves represent the situation for a fixed value of the sparsity parameter d, which varies from $d = 1, \ldots, 10$. The discretization parameter is set to $m' = 3$ for all the cases displayed in this figure.

5.3. Time to solution – dependence on the number of assets

In the first experiment, we study the dependence of the time to solution (TTS) as we vary the number of assets in the portfolio, for both the unary and binary encoding QUBO problems. First for a fixed value of the sparsity parameter d, we vary the number of assets from $n = d, \ldots, 10$, and then repeat this for each value of $d = 1, \ldots, 10$. The discretization parameter is set to $m' = 3$ for all these experiments. The experiment is repeated for all the 10 data sets for each value of n, and the mean and standard deviations of the TTS are calculated. The results for the unary case are plotted in Figure 1, while the results for the binary case are plotted in Figure 2. In these figures, for each value of n and d, we have plotted the mean TTS with the error bars representing the standard deviations of the TTS in each case.

The results obtained are what is expected. It is clearly seen that for a fixed value of d, the TTS increases as we increase the total number of assets in the portfolio n, in fact the increase happens faster than at a linear rate for both the unary and binary problems, with the effect being much more pronounced for the unary case. The effect of increasing the sparsity parameter d seems to have a general effect of increasing the TTS, for a fixed value of n, but this increase does not seem to be monotonic. In fact for the binary case, $d = 3, \ldots, 10$ produces almost identical curves as seen from Figure 2. These effects are intuitively explained by the fact that the TTS should strongly depend on the number of variables involved in the optimization problem, and it is the parameter n that has the most impact on the count of the total number of variables involved in the optimization problem.

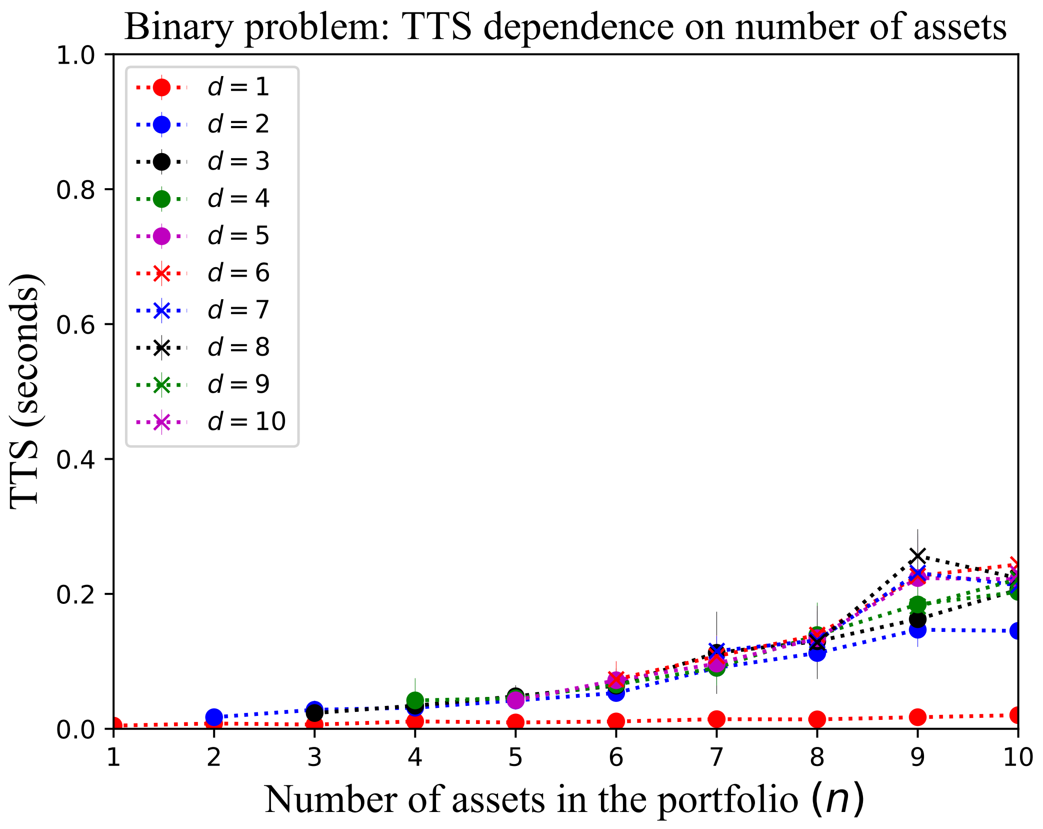

Figure 2: Variation of the time to solution (TTS) with respect to the number of assets n in the portfolio, for the binary encoding QUBO problem (12), where we have plotted the average TTS along with their standard deviation for each value of d and n. Each of the curves represent the situation for a fixed value of the sparsity parameter d, which varies from $d = 1, \dots, 10$. The discretization parameter is set to $m' = 3$ for all the cases displayed in this figure.

Another interesting aspect to note from these figures is the effect that the parameters n and d have on the standard deviation of TTS. Clearly the standard deviation shows an increasing trend as we solve larger problem sizes both in terms of n and d, and the effect is evident for both the unary and binary encoding cases, although it is significantly more pronounced for the unary encoding case. At the moment, we can only offer a speculative explanation of why this should be the case – the branch and bound algorithms employed by CPLEX in searching for the global minimum will perform very differently on different instances of an optimization problem, for the same problem size determined by n, m' and d, as the problem sizes get larger and larger. For some instances the branch and bound algorithm finds the answer easily, while for other instances it may not find it easily, with the variance increasing due to the increase in problem size. Lastly, we should also notice from these results that solving the binary encoding QUBO takes much less time as compared to the unary encoding QUBO, as the number of variables involved in the binary case is much lesser as compared to the unary case, as was already alluded to in previous sections. These numerical results provide quantitative evidence of the theoretical claims.

5.4. Time to solution – dependence on the sparsity parameter

In the next test we wish to study the dependence of TTS as we vary the sparsity parameter d of the portfolio. As in the previous case, we will again fix the parameter controlling the discretization to $m' = 3$. Next we vary the number of assets $n = 1, \dots, 10$, and for each value of n we vary the sparsity parameter $d = 1, \dots, n$, and compute the mean and standard deviations of TTS over the 10 data sets corresponding to each value of n. The results of this experiment are plotted in Figure 3 for the unary encoding QUBO, and in Figure 4 for the binary encoding QUBO.

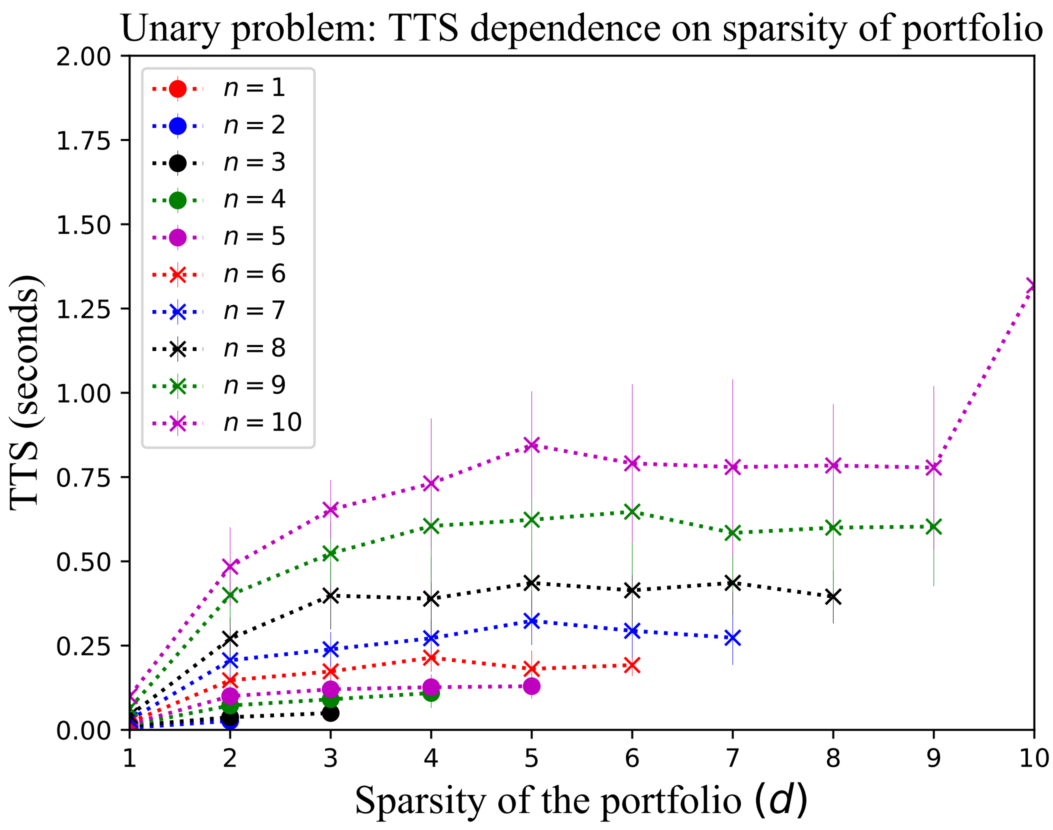

Figure 3: Variation of the time to solution (TTS) with respect to the sparsity parameter d of the portfolio, for the unary encoding QUBO problem (10), where we have plotted the average TTS along with their standard deviation for each value of d and n. Each of the curves represent the situation for a fixed value of the number of assets in the portfolio n, which varies from $n = 1, \dots, 10$. The discretization parameter is set to $m' = 3$ for all the cases displayed in this figure.

The results obtained are somewhat expected from the previous experiment. Here again we clearly see that increasing n completely shifts the variation of TTS with respect to d to a higher value, while for fixed n, the TTS is relatively constant. In fact this conclusion is true for both the unary and binary encoding cases. At the same time, the TTS for the binary encoding case is much lower as compared to the unary encoding case. Again the impact of n on the standard deviation of TTS from these figures is quite noticeable, consistent with the previous experiment.

5.5. Time to solution – dependence on the coarseness of discretization

In the final experiment, we study the dependence of TTS on the parameter m' that controls the coarseness of the discretization of the interval $[0, 1]$. The conclusions presented in this section are for the binary encoding case, but we speculate that similar conclusions will be true even for the unary encoding case. In fact we have numerically verified that the effects are similar for small values of m' upto $m' = 5$ for the unary case on a limited number of runs. The main issue with the unary encoding case is that it uses significantly more number of variables than the binary encoding case, and it leads to very large run times – for example for $n = 10$, $m' = 4$, and $d = 3$, the TTS is around 1 minute, while for $n = 10$, $m' = 5$, and $d = 3$ the run time exceeded 1 hour on a limited number of cases that we tried. As a consequence, getting good estimates for average TTS turned out to be very time consuming for the unary encoding case, and hence we only present the binary encoding case here.

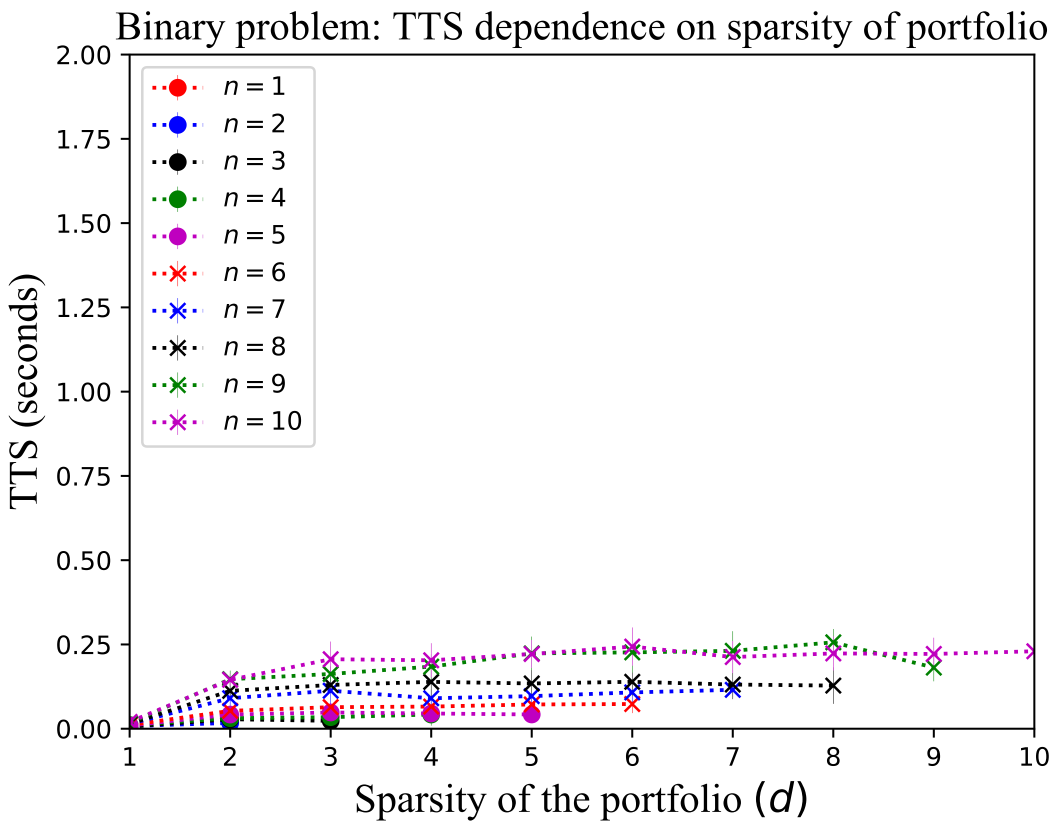

Figure 4: Variation of the time to solution (TTS) with respect to the sparsity parameter d of the portfolio, for the binary encoding QUBO problem (12), where we have plotted the average TTS along with their standard deviation for each value of d and n. Each of the curves represent the situation for a fixed value of the number of assets in the portfolio n, which varies from $n = 1, \ldots, 10$. The discretization parameter is set to $m' = 3$ for all the cases displayed in this figure.

In this experiment we first fix the number of assets n, and then vary the parameter $m' = 1, \ldots, 6$. For each n, the sparsity parameter is set to $d = 3$. Then as in the case of the previous experiments, we compute the mean and standard deviations of the TTS for each value of n and m', over the 10 data sets corresponding to the value of n. Finally we repeat the experiment for different values of $n = 3, \ldots, 8$. The results of the experiment are plotted in Figure 5 as before with error bars representing the standard deviation in TTS. We see clearly that for a fixed value of n, increasing m' leads to a dramatic increase in mean TTS, the rate of increase being clearly greater than any linear function. Increasing n also has a significant impact on TTS, the increase being mostly monotonic for mean TTS, for a fixed value of m'.

6. Conclusion and Discussion

In this paper we have formulated two equivalent QUBO formulations for the single-period index tracking problem with sparsity constraints—the first one based on a unary encoding strategy, and the second one based on a binary encoding strategy. We performed several numerical experiements with both the formulations, and have shown that the binary encoding QUBO outperforms the unary encoding QUBO in terms of solution times for the same problem, and is thus inherently a better way to express the problem. However it should be noted that these conclusions are true in exact arithmetic, that is assuming that CPLEX can exactly solve the unary or binary encoding QUBOs. It is well known that the method of adding penalties to the objective function to convert constrained optimization problems to unconstrained optimization problems is not robust, especially for large problem sizes that either have a lot of constraints or are extremely ill-conditioned, an effect that stems from finite precision computations. The binary encoding formulation is much more susceptible to ill-conditioning than the unary case, and it is expected that for large problem sizes, finite precision solvers

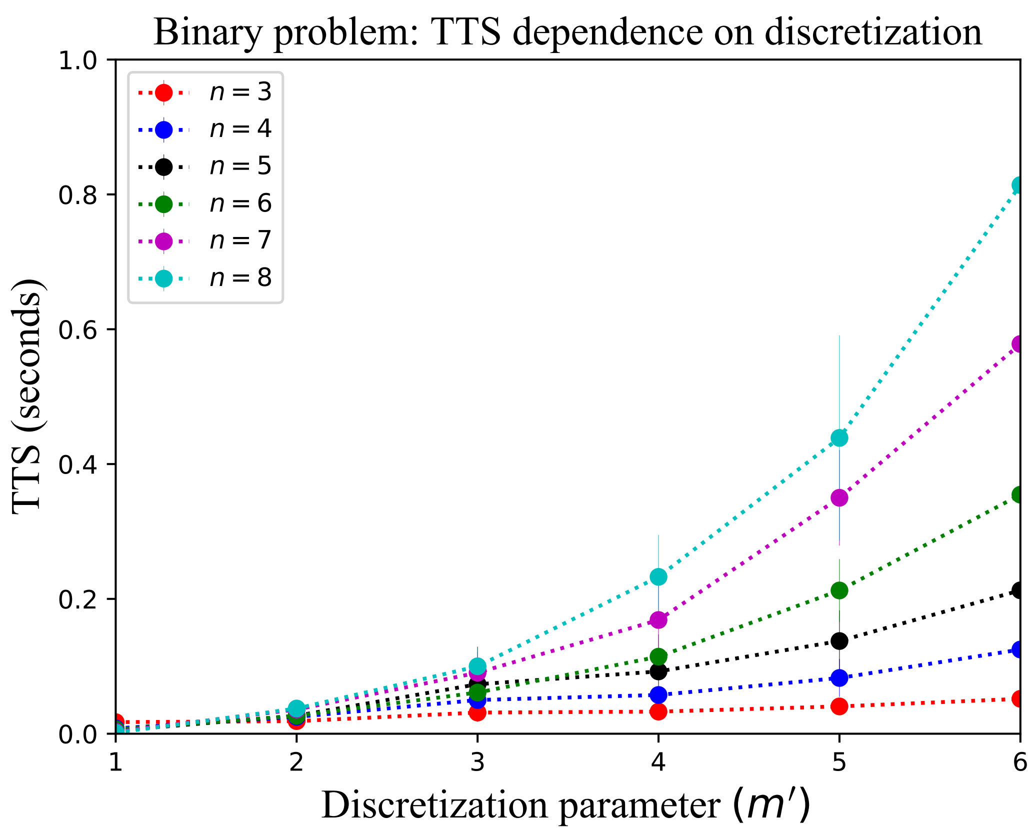

Figure 5: Variation of the time to solution (TTS) with respect to the discretization parameter m', for the binary encoding QUBO problem (12), where we have plotted the average TTS along with their standard deviation for each value of m' and n. Each of the curves represent the situation for a fixed value of the number of assets in the portfolio n, which varies from $n = 2, \ldots, 6$. The sparsity parameter is set to $d = 3$ in each case displayed in the figure.

like CPLEX will perform poorly in terms of being able to solve the problem precisely, that is fail at finding the global minimum, much more than the unary case. Thus there is a trade-off between accuracy and time taken to solve the optimization problems on a finite precision computer that becomes important for large problem sizes.

7. Acknowledgments

We would like to thank Bob Gingrich from the Western Asset Management Company for providing us the data set that was used in this study for performing the numerical experiments.

References

Adachi, S. H., Henderson, M. P., 2015. Application of quantum annealing to training of deep neural networks. arXiv preprint arXiv:1510.06356.

Aharonov, D., Van Dam, W., Kempe, J., Landau, Z., Lloyd, S., Regev, O., 2008. Adiabatic quantum computation is equivalent to standard quantum computation. SIAM review 50 (4), 755–787.

Alidaee, B., Kochenberger, G., Lewis, K., Lewis, M., Wang, H., 2008. A new approach for modeling and solving set packing problems. European Journal of Operational Research 186 (2), 504 – 512. URL `http://www.sciencedirect.com/science/article/pii/S0377221707001889`

Amenc, N., Ducoulombier, F., Goltz, F., Tang, L., 2012. What are the risks of european etfs. EDHEC-Risk Institute Position Paper.

Beasley, J., Meade, N., Chang, T.-J., 2003. An evolutionary heuristic for the index tracking problem. European Journal of Operational Research 148 (3), 621 – 643.
URL `http://www.sciencedirect.com/science/article/pii/S0377221702004253`

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., Lloyd, S., 2017. Quantum machine learning. Nature 549 (7671), 195.

Billionnet, A., Elloumi, S., Jan 2007. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. Mathematical Programming 109 (1), 55–68.
URL `https://doi.org/10.1007/s10107-005-0637-9`

Buckley, I., Korn, R., 1998. Optimal index tracking under transaction costs and impulse control. International journal of theoretical and applied Finance 1 (03), 315–330.

Canakgoz, N., Beasley, J., 2009. Mixed-integer programming approaches for index tracking and enhanced indexation. European Journal of Operational Research 196 (1), 384 – 399.
URL `http://www.sciencedirect.com/science/article/pii/S037722170800283X`

Childs, A. M., Farhi, E., Goldstone, J., Gutmann, S., 2000. Finding cliques by quantum adiabatic evolution. arXiv preprint quant-ph/0012104.

Childs, A. M., Farhi, E., Preskill, J., 2001. Robustness of adiabatic quantum computation. Physical Review A 65 (1), 012322.

Clarke, R. G., Krase, S., Statman, M., 1994. Tracking errors, regret, and tactical asset allocation. The Journal of Portfolio Management 20 (3), 16–24.

Coleman, T. F., Li, Y., Henniger, J., 2006. Minimizing tracking error while restricting the number of assets. The Journal of Risk 8 (4), 33.

Connor, G., Leland, H., 1995. Cash management for index tracking. Financial Analysts Journal 51 (6), 75–80.
URL `http://www.jstor.org/stable/4479886`

Farhi, E., Goldstone, J., Gutmann, S., 2014. A quantum approximate optimization algorithm. arXiv preprint arXiv:1411.4028.

Farhi, E., Goldstone, J., Gutmann, S., Lapan, J., Lundgren, A., Preda, D., 2001. A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. Science 292 (5516), 472–475.

Farhi, E., Goldstone, J., Gutmann, S., Sipser, M., 2000. Quantum computation by adiabatic evolution. arXiv preprint quant-ph/0001106.

Farhi, E., Gosset, D., Hen, I., Sandvik, A., Shor, P., Young, A., Zamponi, F., 2012. Performance of the quantum adiabatic algorithm on random instances of two optimization problems on regular hypergraphs. Physical Review A 86 (5), 052334.

Gottesman, D., 1998. Theory of fault-tolerant quantum computation. Physical Review A 57 (1), 127.

Guastaroba, G., Speranza, M., 2012. Kernel search: An application to the index tracking problem. European Journal of Operational Research 217 (1), 54 – 68.
URL `http://www.sciencedirect.com/science/article/pii/S0377221711008071`

Hammer, P. L., Shlifer, E., Mar 1971. Applications of pseudo-boolean methods to economic problems. Theory and Decision 1 (3), 296–308.
URL `https://doi.org/10.1007/BF00139572`

Hansen, P., 1979. Methods of nonlinear 0-1 programming. Annals of Discrete Mathematics 5, 53–70.

Jansen, R., Van Dijk, R., 2002. Optimal benchmark tracking with small portfolios. The journal of portfolio management 28 (2), 33–39.

Johnson, M. W., Amin, M. H., Gildert, S., Lanting, T., Hamze, F., Dickson, N., Harris, R., Berkley, A. J., Johansson, J., Bunyk, P., et al., 2011. Quantum annealing with manufactured spins. Nature 473 (7346), 194.

Jordan, S. P., Farhi, E., Shor, P. W., 2006. Error-correcting codes for adiabatic quantum computation. Physical Review A 74 (5), 052322.

Kadowaki, T., Nishimori, H., 1998. Quantum annealing in the transverse ising model. Physical Review E 58 (5), 5355.

Karlow, D., 2012. Comparison and development of methods for index tracking. Ph.D. thesis, Frankfurt School of Finance and Management.

Karp, R. M., 1972. Reducibility among combinatorial problems. Complexity of Computer Computations, 85–103.

Katayama, K., Narihisa, H., 2001. Performance of simulated annealing-based heuristic for the unconstrained binary quadratic programming problem. European Journal of Operational Research 134 (1), 103 – 119.
URL `http://www.sciencedirect.com/science/article/pii/S0377221700002423`

Kochenberger, G., Hao, J.-K., Glover, F., Lewis, M., Lü, Z., Wang, H., Wang, Y., Jul 2014. The unconstrained binary quadratic programming problem: a survey. Journal of Combinatorial Optimization 28 (1), 58–81.
URL `https://doi.org/10.1007/s10878-014-9734-0`

Kurihara, K., Tanaka, S., Miyashita, S., 2009. Quantum annealing for clustering. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. AUAI Press, pp. 321–328.

Laughhunn, D. J., 1970. Quadratic binary programming with application to capital-budgeting problems. Operations Research 18 (3), 454–461.
URL `https://doi.org/10.1287/opre.18.3.454`

Levit, A., Crawford, D., Ghadermarzy, N., Oberoi, J. S., Zahedinejad, E., Ronagh, P., 2017. Free energy-based reinforcement learning using a quantum processor. arXiv preprint arXiv:1706.00074.

Lewis, M., Kochenberger, G., Alidaee, B., 2008. A new modeling and solution approach for the set-partitioning problem. Computers & Operations Research 35 (3), 807 – 813, part Special Issue: New Trends in Locational Analysis.
URL `http://www.sciencedirect.com/science/article/pii/S0305054806001201`

Lobo, M. S., Fazel, M., Boyd, S., 2007. Portfolio optimization with linear and fixed transaction costs. Annals of Operations Research 152 (1), 341–365.

Lucas, A., 2014. Ising formulations of many np problems. Frontiers in Physics 2, 5.
URL `http://journal.frontiersin.org/article/10.3389/fphy.2014.00005`

Markowitz, H. M., 1989. Mean-variance analysis in portfolio choice and capital markets. B. Blackwell.

Naumenko, K., Chystiakova, O., 2015. An empirical study on the differences between synthetic and physical etfs. International Journal of Economics and Finance 7 (3), 24.

Neukart, F., Compostella, G., Seidel, C., von Dollen, D., Yarkoni, S., Parney, B., 2017. Traffic flow optimization using a quantum annealer. Frontiers in ICT 4, 29.

Neukart, F., Von Dollen, D., Seidel, C., Compostella, G., 2018. Quantum-enhanced reinforcement learning for finite-episode games with discrete state spaces. Frontiers in Physics 5, 71.

Nocedal, J., Wright, S. J., 2006. Numerical Optimization. Springer.

Pardalos, P. M., Jha, S., 1992. Complexity of uniqueness and local search in quadratic 0–1 programming. Operations Research Letters 11 (2), 119 – 123.
URL `http://www.sciencedirect.com/science/article/pii/0167637792900433`

Preskill, J., 1998. Fault-tolerant quantum computation. In: Introduction to quantum computation and information. World Scientific, pp. 213–269.

Pudenz, K. L., Albash, T., Lidar, D. A., 2014. Error-corrected quantum annealing with hundreds of qubits. Nature communications 5, 3243.

Pudenz, K. L., Albash, T., Lidar, D. A., 2015. Quantum annealing correction for random ising problems. Physical Review A 91 (4), 042302.

Roll, R., 1992. A mean/variance analysis of tracking error. The Journal of Portfolio Management 18 (4), 13–22.

Rosenberg, G., Haghnegahdar, P., Goddard, P., Carr, P., Wu, K., De Prado, M. L., 2016. Solving the optimal trading trajectory problem using a quantum annealer. IEEE Journal of Selected Topics in Signal Processing 10 (6), 1053–1060.

Rudolf, M., Wolter, H.-J., Zimmermann, H., 1999. A linear model for tracking error minimization. Journal of Banking & Finance 23 (1), 85–103.

Ruiz-Torrubiano, R., Suárez, A., 2009. A hybrid optimization approach to index tracking. Annals of Operations Research 166 (1), 57–71.

Santoro, G. E., Martoňák, R., Tosatti, E., Car, R., 2002. Theory of quantum annealing of an ising spin glass. Science 295 (5564), 2427–2430.

Sato, I., Tanaka, S., Kurihara, K., Miyashita, S., Nakagawa, H., 2013. Quantum annealing for dirichlet process mixture models with applications to network clustering. Neurocomputing 121, 523–531.

Shapcott, J., 1992. Index tracking: genetic algorithms for investment portfolio selection. Edinburgh Parallel Computing Centre, EPCC–SS92–24.

Shor, P. W., 1996. Fault-tolerant quantum computation. In: Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on. IEEE, pp. 56–65.

Sorenson, E., Miller, K., Samak, V., 1998. Allocating between active and passive management. Financial Analysts Journal 54 (5), 18–31.

Strub, O., Baumann, P., 2018. Optimal construction and rebalancing of index-tracking portfolios. European Journal of Operational Research 264 (1), 370 – 387.
URL `http://www.sciencedirect.com/science/article/pii/S0377221717305969`

Wang, M., Xu, C., Xu, F., Xue, H., 2012. A mixed 0–1 lp for index tracking problem with cvar risk constraints. Annals of Operations Research 196 (1), 591–609.