

# Approaching Collateral Optimization for NISQ and Quantum-Inspired Computing

Megan Giron,<sup>1,\*</sup> Georgios Korpas,<sup>1,2</sup> Waqas Parvaiz,<sup>1</sup> Prashant Malik,<sup>3</sup> and Johannes Aspman<sup>2</sup>

<sup>1</sup>*HSBC Lab, Innovation & Ventures, 8 Canada Square, London, E14 5HQ, U.K.* <sup>†</sup>

<sup>2</sup>*Department of Computer Science, Czech Tech. University in Prague, Karlovo nám. 13, Prague 2, Czech Republic*

<sup>3</sup>*Markets and Security Services, 8 Canada Square, London, E14 5HQ, U.K.*

(Dated: May 29, 2023)

Collateral optimization refers to the systematic allocation of financial assets to satisfy obligations or secure transactions, while simultaneously minimizing costs and optimizing the usage of available resources. This involves assessing number of characteristics, such as cost of funding and quality of the underlying assets to ascertain the optimal collateral quantity to be posted to cover exposure arising from a given transaction or a set of transactions. One of the common objectives is to minimise the cost of collateral required to mitigate the risk associated with a particular transaction or a portfolio of transactions while ensuring sufficient protection for the involved parties. Often, this results in a large-scale combinatorial optimization problem. In this study, we initially present a Mixed Integer Linear Programming (MILP) formulation for the collateral optimization problem, followed by a Quadratic Unconstrained Binary optimization (QUBO) formulation in order to pave the way towards approaching the problem in a hybrid-quantum and NISQ-ready way. We conduct local computational small-scale tests using various Software Development Kits (SDKs) and discuss the behavior of our formulations as well as the potential for performance enhancements. We further survey the recent literature that proposes alternative ways to attack combinatorial optimization problems suitable for collateral optimization.

## I. INTRODUCTION

In the context of a financial transaction, wherein one party lends assets to another, the lender assumes a credit risk arising from the possibility that the counterparty may default on their obligations. This risk also arises in derivatives transactions where the party “in-the-money” is exposed to the party “out-of-the-money”. To mitigate this risk, the borrower is required to provide low-risk securities (such as cash, bonds, or equities) to the lender for the duration of the transaction. This practice, known as *collateralization* [1], serves as a form of security against loan defaults as the lender can seize these assets to offset any losses resulting from default. The value of the collateral received is expected to be commensurate with the outstanding exposure, in order to effectively counterbalance the associated risk.

Often, a bilateral contract (or schedule) is formed to agree on the terms under which securities can be considered collateral, the process of evaluating the value of these assets, and other regulations. The relevant party may then accordingly select the assets they post to the counter-party. For large financial institutions, such as banks, this can involve a pool of numerous assets to choose from which need to be distributed amongst a portfolio of various counterparties (other banks, hedge funds, central banks, etc.). Each asset has an associated opportunity cost, which is a measure of how valuable the asset would be if it were used for another purpose. As well as a cost related to the risk of posting to a particular counterparty amongst other administrative costs.

The bank must therefore carefully consider their choice of transactions to reduce these costs. However, the magnitude of the possible combinations of allocations for large institutes makes this a time-consuming process. In addition, poor collateral management can have significant consequences. The 2008 financial crisis was partly due to collateralization of high-risk securities and led to the bankruptcy of some of the largest financial institutions, among other consequences [2].

This crisis has led to the reformation of many financial processes through the implementation of regulations such as Basel III [3], Dodd-Frank Act [4], EMIR [5] as well as motivating academic research with regards to how collateral can be better managed [6, 7]. These studies relating to collateral management are generally centered on financial theories such as risk aversion and its global financial impact. A crucial aspect of collateral management is the development of an automated process that selects the optimal allocations. Despite the importance of collateral optimization (ColOpt), the literature surrounding this topic is sparse due to the competitive advantage these strategies offer to financial institutions.

Naturally, linear programming algorithms can provide a framework for tackling such problems [8]. Specifically, ColOpt is suitable to be implemented using mixed-integer linear program (MILP) solvers such as the ones available with IBM CPLEX [9], Gurobi [10], or Mosek [11]. The success of a given ColOpt instance, or the quality of its solution, is dependent on a clear mapping between the business problem and the mathematical formulation as well as the choice of the precise implementing algorithm. The benefit of using numerical optimization is that we attain the allocation selections in a single process, which is in contrast to other proposed models such as “ranking-based”, “economic-cost” and “waterfall” models which

\* Corresponding author

<sup>†</sup> megan.giron@hsbc.com

are sequential in nature, rather than automated [12]. However, there are several limitations of MILP solvers when applied to problems such as ColOpt, which potentially involve complex nonlinearities and large-scale datasets. For example, MILP solvers have exponential worst-case complexity and can take a significant amount of time to solve large-scale complex optimization problems. While having convergence certificates is very desirable, the exponential complexity is often a problem for many ColOpt instances that involve a large number of decision variables and constraints and it is not uncommon to either have long solution times or even infeasibility. That said, MILP solvers are the standard in both industrial and academic applications but the community is very keen on exploring alternative approaches.

Different avenues to (approximately) solve such computationally challenging problems could be provided through alternative computational models. For example, in Ref. [13], IBM’s True Spike neuromorphic computer [14] was utilized to find approximate solutions for the graph partitioning problem. More interest is directed towards quantum computing [15]. These computers rely on quantum-mechanical effects for storing and processing information. However, because of the fragile environment required for these effects to occur, the realization of fault-tolerant quantum computers is still a difficult task. Despite this, it is believed that Noisy Intermediate-Scale Quantum (NISQ) era devices can provide an advantage in the finance industry since these business use cases can be well formulated for near-term quantum devices [16, 17]. The field of “quantum finance” can be divided into three sections: stochastic modeling (for example, quantum alternatives to Monte Carlo simulations) [18–22], machine learning [18–20, 23–26] and optimization [18, 24, 27–33], all of which have had a recent gain in interest.

Quantumly, there are many approaches to follow in order to approximate better solutions for a variety of NP-Hard problem instances encountered in finance. The main three approaches have a common theme: conversion of the original mathematical formulation of the problem from a linear program (LP) formulation to a quadratic unconstrained binary optimization problem (QUBO). The reason lies in the inherent ability of quantum or hybrid approaches by modeling the Ising model type of systems (see App. B). There, an optimization problem is mapped to the classical Hamiltonian of the Ising model, where its ground state encodes the optimum. As a matter of fact, many NP-Hard problems, including Karp’s list of 21 NP-Hard problems, are known to admit at least one formulation of the Ising model [34]. The QUBO or Ising approach can be used and problems can be mainly tackled as follows:

- Using variational quantum algorithms (VQAs) [35] such as the Quantum Approximate Optimization Algorithm (QAOA) [36] on gate-based quantum computers (e.g., IBM’s superconducting quantum computer). A variety of tests have been performed in this context with significant (qubit) resource im-

provements recently [37].

- Using quantum annealing (QA) [38–42] on adiabatic quantum computers (quantum annealers) such as the D-Wave hardware [43] (see [44] for an application to portfolio optimization).
- Using quantum-inspired methods which can be understood as using the QUBO formulation of the problem of interest with any approach that ranges from simulated annealing [45] on high-performance clusters or digital annealing [46] such as Fujitsu’s FPGA-based “quantum-inspired” classical hardware DAU [47].

These approaches are very promising and it is widely expected that near-term quantum computers may have a good chance to provide computational advantage [48, 49].

Additionally, there exists evidence that the class of problems that NISQ computers can solve is not a subset [50] of BPP[51]. However, we realize that their heuristic nature can conceal their applicability, especially when considering problem instances with sizes suitable for high-quality MILP solvers. Such a result was reported in Ref. [52] wherein the authors discuss how D-Wave’s 2000Q machine (as well as classical simulated annealing) failed to even come close to the branch and bound approaches [53, 54] for solving certain instances of the Knapsack Problem (KnapsackProb) (see Sec. II). One can potentially try to use the VQA approach instead. In this context, Nannicini [55] reported that there seems to be a lack of transparent computational advantage in introducing entanglement when VQAs are used. This lack of computational advantage is expected to some extent due to the well-known local minima problem that VQAs exhibit [56] (see [57] for the proposed way to avoid this problem). In addition, bias in the noise of circuits that implement VQAs can unfavourably affect the convergence ratios [58]. Other limitations are discussed in [59].

However, performance advantages have been showcased in a variety of applications within the context of digital and quantum annealing solutions, tensor networks, and analog and digital (gate-based) quantum computing. For example, Ref. [60] studied the maximal independent set (MaxIndSet) problem and found a superlinear quantum speedup, as opposed to classical solutions, when considering very hard graphs. From the point of view of computational complexity, the MaxIndSet problem is not particularly different from other NP-Hard problems such as the KnapsackProb or other NP-Hard problems that admit a MILP formulation, and as such the results of [60] are encouraging for other problems as well. It is worth mentioning, that D-Wave recently announced [61] the largest quantum simulation done, in different context (spin glasses) to what is of interest here.

In this paper, we study the ColOpt problem in detail and provide a MILP formulation that we use as a testbed for: (i) formulating a QUBO version of ColOpt which

makes it suitable for feeding into quantum and quantum-inspired solvers and (ii) performing small scale simulations of such solvers and comparing to MILP. Specifically, in our MILP formulation we choose our objective to be minimization of the cost of posting collateral (different approaches to the objective of ColOpt have been proposed, for example, see [12]). We survey and try numerous QUBO encodings and find that, modulo emulator limitations, in such small instances the quantum-inspired formulations perform well enough to be promising for implementation on real quantum or quantum-inspired hardware for very large instances. The structure of the paper is as follows. In Sec. II we provide an overview of some of the different QUBO formulations for the KnapsackProb problem. This section serves both as an introduction to the concepts used throughout the paper and to inform the ColOpt problem that follows. In Sec. III we provide the MILP formulation of the ColOpt problem as well as a few QUBO proposals. In Sec. IV we provide a few numerical results using the formulation of the previous section. Finally, we summarize and conclude in Sec. V.

We want to clarify that while our paper investigates various QUBO formulations for the KnapsackProb problem, our ultimate goal is to apply this information to the ColOpt problem. Specifically, we plan to use the best-performing QUBO formulations from our KnapsackProb study to formulate and solve the collateral optimization problem using QUBO. However, we would like to note that our paper does not aim to provide an empirical comparison between quantum and classical approaches for solving MILPs, given the limited computational resources available to us (see [62] for work on the comparison of classical and quantum (adiabatic) optimization, wherein the authors discovered “surprising” results favoring the D-Wave machine). Additionally, we focus on small problem instances only, and we acknowledge that all the results presented in our study are heuristic in nature. Nevertheless, based on the literature results on the potential of QUBO formulations (quantumly and not only), we believe that the formulations we propose may have value in tackling larger instances of the collateral optimization problem and therefore may warrant further investigation.

In summary, the main objective of our paper is to present a case study on the formulation and approach of the ColOpt problem using quantum computing techniques, with the overarching aim of advancing the ongoing effort towards achieving “quantum advantage” in practical applications.

## II. INTERLUDE WITH THE KNAPSACK PROBLEM

To inform and ensure that our formulations and the subsequent computations, we perform a simple test using a small KnapsackProb instance. In essence,

Weight	23	31	29	44	53	38	63	85	89	82
Value	92	57	49	68	60	43	67	84	87	72

TABLE I: Input data used of the specific KnapsackProb instance considered in this paper. The total capacity of the knapsack is 165.

KnapsackProb involves determining the optimal approach to filling a knapsack of capacity  $W$  with the highest possible value from a set of  $n$  items that have specific sizes and corresponding values (see Table I). This problem is of interest due to its simplicity to formulate, and its simple constraints. For us, it is further interesting since we view the ColOpt problem as a (somewhat complicated) generalization.

Given the large number of “hard” constraints of the ColOpt problem, see Sec. III, we aim to compare formulation for small instances of the Knapsack Problem with the hope to inform our approach for collateral optimization. The “standard route” to encode constraints to a QUBO model is by using (balanced) slack variables for penalization [63]. A different approach is using “unbalanced” penalization [64] which turns out to be particularly useful for QAOA solutions as it reduces the resources required by a gate-based machine.

The MILP formulation of the KnapsackProb is a well-known and straightforward approach. In the problem instance we consider, we are given a set of weights  $w \in \mathbb{Z}_{\geq 0}^n$  and their corresponding values  $v \in \mathbb{Z}_{\geq 0}^n$ , and the objective is to maximize the total value of the items that can be packed into a knapsack subject to a given weight limit. The problem can be mathematically defined as follows:

$$\begin{aligned} \max \quad & \sum_{i=1}^n v_i x_i, \\ \text{s.t.} \quad & \sum_{i=1}^n w_i x_i \leq W, \end{aligned} \tag{1}$$

where  $W$  is the maximum weight limit (threshold) of the knapsack and  $x_i$  is the binary variable representing whether the  $i$ -th item is to be placed in the knapsack. The best running-time algorithm for solving the KnapsackProb is based on dynamic programming with pseudo-polynomial complexity  $\mathcal{O}(d_n W)$  [65], where  $d_n$  is the number of distinct weights available while near-linear running times algorithms (in  $d_n, W$ ) were documented in [66].

The specific problem instance we consider in our study, as outlined in [67], comprises ten items and possesses a known optimal solution. We leverage this knowledge to heuristically evaluate the effectiveness of our approach and guide our efforts towards tackling the larger ColOpt problem discussed in Section III. The relevant (toy) data pertaining to the items in this problem instance can be found in Table I.

Although the KnapsackProb is known to be (weakly) NP-complete, simple instances such as the one we consider in this study can be efficiently solved by a range of

classical solvers. For our experimental analysis, we used the HiGHS [68] and GLPK [69] solvers, both of which yielded solutions that were in agreement with the known optimal solution of the problem instance, as expected, while for the QUBO-formulated problem, we tested the open-source Julia libraries ToQUBO.jl [70], Qiskit’s optimization module [71], the open-source Python library PyQubo [72] (in both cases operating under simulated annealing) and the proprietary Digital Annealer of Fujitsu [47].

### A. Quadratic Unconstrained Binary optimization

The QUBO model can be applied to a wide range of combinatorial optimization problems that are known to be NP-hard, such as the maximum cut, minimum vertex cover, multiple knapsack, and graph coloring problems. Its applications span a diverse set of domains, including the automotive industry [73], portfolio optimization [74–76], traffic flow optimization [77], job scheduling [78–80], railway conflict management [81], bioinformatics [82], and others [83]. An extensive list of QUBO formulations for interesting problems can be found in [84].

Due to its one-to-one mapping to the Ising Hamiltonian, QUBOs have become a fundamental element of quantum-inspired computing. Both the Digital Annealer developed by Fujitsu and the adiabatic quantum computers manufactured by D-Wave Systems (as well as other vendors such as Qilimanjaro) employ the QUBO model to address complex optimization problems. Additionally, several approaches based on tensor networks seem to be suitable for a variety of QUBO-formulated optimization problems [85–88]. Although QUBO is particularly well-suited to these technologies, it can also be employed in NISQ devices using algorithms such as the QAOA. As such, the QUBO model is an important tool for quantum optimization with potential applications across a range of quantum computing platforms, and formulating constrained problems as such highly affects the quality of the solutions obtained.

Let us summarise the basics of QUBO via a graph problem. Given an undirected graph  $G = (V, E)$  with a vertex set  $V = \{1, 2, \dots, N\}$  connected by the edge set  $E = \{(i, j), \}, i, j \in V$ , the cost function is defined as:

$$\min \sum_{i=1}^N A_{ii}x_i + \sum_{i=1}^{N-1} \sum_{j>i}^N A_{ij}x_i x_j, \quad (2)$$

where  $x \in \{0, 1\}$  are the binary variables and the elements  $A_{ij} \in \mathbb{R}^{N \times N}$  are the problem instance parameters.

At its most fundamental level, a QUBO can be expressed as:

$$\min \quad x^T Q x + b, \quad (3)$$

where the matrix  $Q \in \mathbb{R}^{N \times N}$  contains the problem instance and  $b \in \mathbb{R}$  is a constant offset term.

By using a suitable change of variables  $x_i = \frac{1-\sigma_i}{2}$ , Eq.(2) can be mapped onto the Ising Model Hamiltonian as:

$$H(s) = - \sum_j h_j \sigma_j - \sum_{j<k} J_{jk} \sigma_j \sigma_k, \quad (4)$$

where  $\sigma \in \{-1, 1\}^N$  are the (classical) spins,  $h \in \mathbb{R}^N$  is the magnetic field, and  $J \in \mathbb{R}^{N \times N}$ ,  $\text{diag}(J) = 0$ , the spin-spin interaction symmetric matrix between adjacent spins  $j$  and  $k$ . See App. B for more details.

For the QUBO formulation of **KnapsackProb** we can take several slack-based approaches including “off-the-shelf” LP-to-QUBO converters such as Qiskit’s `QuadraticProgramToQubo` class and methods as well as the Julia package `ToQUBO.jl`. Further to that, we can perform a “custom” slack formulation and also use the unbalanced penalization method we mentioned previously.

### B. Slack Variable Formulation

In the process of converting MILPs to QUBOs, it is common practice to introduce a slack variable,  $S \in \mathbb{R}_{\geq 0}$  (whose purpose will be discussed shortly), for each linear inequality and transform it into an equivalent linear equality. Subsequently, a penalty term is constructed based on the slack variable, and the term is squared as per the standard approach, as outlined in [63] (see also [89]).

A variety of different slack-based QUBO formulations exist for the **KnapsackProb** [90]. Here, the corresponding penalisation term with weight  $\lambda_0 \in \mathbb{R}_+$  is given by the equality:

$$\lambda_0 \left( \sum_{i=1}^n w_i x_i - W + S \right)^2 = 0. \quad (5)$$

The purpose of the auxiliary slack variable  $S$  is to reduce this term to 0 once the constraint has been satisfied,  $0 \leq S \leq \max_x \sum_i^n w_i x_i - W$ . In practice,  $S$  is decomposed into binary representation using variables  $s_k \in \{0, 1\}$  as follows:

$$S = \sum_{k=1}^{N_s} 2^{k-1} s_k. \quad (6)$$

The parameter  $N_s$  corresponds to the number of binary variables required to represent the maximum value that can be assigned to the slack variable, and in the case of **KnapsackProb**,  $N_s = \lceil \log_2(W) \rceil$ , where  $\lceil x \rceil$  is the ceiling function. Formulating the slack variable as in Eq. (6) is commonly referred to as the “log-encoding”.

The full QUBO formulation for the **KnapsackProb** takes the form of maximizing the objective function:

$$\sum_i^n v_i x_i - \lambda_0 \left( \sum_i^n w_i x_i - W + \sum_{k=1}^{N_s} 2^{k-1} s_k \right)^2, \quad (7)$$

which can be understood as an augmented Lagrangian [91]. Alternatively, there are other QUBO formulations of the `KnapsackProb` that we could decide to use that follow a similar Lagrangian paradigm. For example, we can instead consider maximizing the following objective function:

$$\sum_i^n v_i x_i - \lambda_0 \left( \sum_i^n w_i x_i - \sum_{k=1}^W k s_k \right)^2 - \lambda_1 \left( 1 - \sum_{k=1}^W s_k \right)^2. \quad (8)$$

In this formulation, known as “one-hot encoding”, the number of slack bits is equal to the capacity of the knapsack,  $W$ . Here, an additional penalty term is required to enforce only one of these slack bits to be assigned a value of 1. A drawback of this formulation is that the binary input length for the slack variables scales linearly with the values of the constraints, hence it can lead to an unreasonably large number of bits for problem instances with large  $W$  which can exhaust available resources. This issue becomes quite relevant in Sec. IV.

### C. Balanced Slack-Based Approaches

In this section, we give an overview of the approaches used to determine solutions for different balanced formulations of the `KnapsackProb` QUBO. The known optimal solution for our small instance that we consider corresponds to an objective value of 309 and uses the full capacity of the knapsack.

**Off-The-Shelf Converters.** The first “off-the-shelf” approach we tried is `ToQUBO.jl`, an open-source Julia package that automatically reformulates a variety of optimization problems, including MI(L)Ps, to a QUBO. The user can use the `JuMP` [92] package to build the MILP form of `KnapsackProb`. `ToQUBO.jl` provides 6 ways for encoding variables into binary representations. This includes the logarithmic and one-hot approaches mentioned above as well as other less well-known techniques. For continuous decision variables, this can be very useful since the user can provide a tolerance factor to manage the upper bound on the representation error caused by the binarization. Additionally, `ToQUBO.jl` works in conjunction with `QUBODrives.jl`, a companion package that provides common API to use QUBO sampling and annealing machines such as D-Wave’s simulated annealer and, with license, the quantum annealer via `DWave-Neal.jl`. We make use of `ToQUBO.jl` to employ both of the aforementioned binary encodings, both successfully finding the optimal solution. However, since the unbalanced QUBO formulation is a recent development which has not been widely adopted, hence its encoding is not available as part of the `ToQUBO.jl` package.

Another “off-the-shelf” converter is provided by Qiskit’s optimization module which includes functionality for automatically transforming quadratic programs

into QUBOs (the binary property allows us to use such). This transformation can be accomplished by first initializing a `QuadraticProgram` and subsequently utilizing the `QuadraticProgramToQubo` class to convert it into a QUBO via the log-encoding method for slack variables. The module allows the formulated QUBO to feed into several algorithms used by Qiskit to solve optimization problems, such as `SamplingVQE` or `QAOA`, however, the user can also extract the coefficient matrix to use with other solvers. We input this coefficient matrix into `neal` and Fujitsu’s Digital Annealer, where we found that both solvers are able to reach optimum, with the caveat that a large number of runs are needed compared to other methods. For larger problem instances, this can become computationally expensive and thus may be an inadequate choice for `ColOpt`.

There are several variations of the balanced slack-based approach to QUBO formulations, which we summarize now.

**Log-encoding.** This approach refers to implementing Eq. (7). The number of slack bits required for this approach, for the instance of interest, is  $N_s = 8$ . Two different regimes for the weight of the penalty term,  $\lambda_0$ , are checked:

- (a) where the penalty term and the cost function have equal weighting ( $\lambda_0 = 1$ ), and
- (b) where the penalty term is more important than the cost function ( $\lambda_0 = 1 \times 10^4$ ).

We utilized the `neal` package to implement D-Wave’s simulated annealer as our heuristic optimizer in both scenarios. The simulated annealer successfully returned the optimal solution in both cases, which was consistent with the results obtained from the classical solvers. The (emulation of the) Digital Annealer from Fujitsu was employed for both scenarios as well. Similar to the case with `neal`, it successfully produced optimal solutions in both instances.

**One-hot encoding.** We repeated the previously described process but with a focus on analyzing the solutions of Eq. (8). As previously mentioned, this type of formulation requires a large number of bits to encode the slack variables for large knapsack capacities, in this case, 165 bits. In this approach, we once more explored various weight regimes for  $\lambda_0$  and  $\lambda_1$ , in relation to the weight of the cost function term. By selecting  $\lambda_0 = 10^{-1}$  and  $\lambda_1 = 10^3$ , we consistently identified the optimal solution for both `neal` and Fujitsu cases.

### D. Unbalanced Penalization Approach

Given that the number of qubits required scales proportionally with the number of variables, we sought to employ the methodology proposed by Montanez-Barrera *et al.* [64] in order to eliminate the need for slack variables. To accomplish this, we adopted an approximation

technique that creates penalty terms that take on small values when the constraint is fulfilled and large values when it is violated. We rearrange the inequality to define an auxiliary function:

$$h(x) = \sum_i^n w_i x_i - W \leq 0. \quad (9)$$

Using the exponential function  $f(x) := e^{h(x)}$ , the constraint on  $h(x)$  is satisfied. However, since only linear and quadratic terms may be encoded into a QUBO, it is necessary to use a 2<sup>nd</sup> order Taylor approximation of  $f(x)$ . For weights  $\lambda_0$  and  $\lambda_1$ , the resulting QUBO for the KnapsackProb problem reads:

$$\sum_i^n v_i x_i + \lambda_0 \left( \sum_i^n w_i x_i - W \right) + \lambda_1 \left( \sum_i^n w_i x_i - W \right)^2 = 0. \quad (10)$$

It should be noted that Ref. [64] investigated this type of QUBO formulation for different instances of the Traveling Salesperson Problem (TravSalProb), Bin Packing Problem (BinPackProb), and KnapsackProb and found that the minimum energy eigenvalue of the corresponding Hamiltonian did not necessarily coincide with the optimal solution. However, the optimal solution was always found to be amongst the lowest-energy eigenvalues which enhances the confidence for this approach in large-scale experiments.

Using the weights provided by Ref. [64], as a quick non-fine-tuned check, we were able to reach optimality for this KnapsackProb instance for both PyQUBO and Fujitsu unbalanced approaches, signifying that the unbalanced formulation exhibits (some) robustness. For larger KnapsackProb instances tested we produced close to optimal results which, however, would periodically softly break the maximum weight limit.

### E. Summary of the simulations

Below, in Table II, we provide a list of MILP and QUBO solvers, using the two aforementioned approaches, slack-based and unbalanced.

### F. Survey of Alternative Approaches

There exist various alternative approaches that exhibit varying degrees of divergence, in terms of algorithmic implementation, from the aforementioned methods. This section serves solely to provide a survey of some of these approaches without undertaking an experimental analysis.

**Quantum Hybrid Frank-Wolfe method.** Recently, a hybrid quantum generalization to the Frank-Wolfe

Problem Encoding	Reference	Solver
ILP	[69]	GLPK
	[68]	HiGHS
ToQUBO.jl	[70]	D-Wave
QuadraticProgramToQUBO	[71]	D-Wave (PyQubo) Fujitsu
Log	Sec. II C	D-Wave (PyQubo) Fujitsu
One-hot	Sec. II C	D-Wave (PyQubo) Fujitsu
Unbalanced	Ref. IID, [64]	D-Wave (PyQubo) Fujitsu

TABLE II: Summary of methods used to solve the provided instance of the KnapsackProb. All methods found the optimal solution.

method was proposed in [93]. This hybrid (Quantum) Frank Wolfe (Q-FW) augmented Lagrangian method is suitable to tackle large QUBO instances due to a tight copositive relaxation of the original QUBO formulation while dealing with the expensive hyper-parameter tuning found in other QUBO heuristics. The Q-FW method first formulates constrained QUBOs as copositive programs, then employs the Frank-Wolfe method while satisfying linear (in)equality constraints. This is converted to a set of unconstrained QUBOs suitable to be run on, e.g., quantum annealers. It was found that Q-FW successfully satisfied linear equality and inequality constraints, in the context of QUBOs in computer vision applications and the authors of [93] solved intermediary QUBO problems on actual quantum devices demonstrating that Q-FW offers a promising alternative the validity of Q-FW, to “traditional” quantum QUBO solvers.

**Grover adapted binary optimization.** Moving to Fault-Tolerant architectures, quadratic speed-up for combinatorial optimization problems is achievable with the Grover adapted binary optimization (GABO) [94] when compared to brute force search. However, to achieve this, efficient oracles must be developed to represent problems and identify states that satisfy specific search criteria. Quantum arithmetic is commonly utilized to accomplish this task, but this approach can be expensive in terms of required Toffoli gates and ancilla qubits, which may pose a challenge in the near future. Interestingly [94] provides such an oracle construction which makes GABO a promising approach for future quantum computers.

**Graph Neural Networks.** Physics-inspired Graph Neural Networks (GNNs) have been used in [95] (see [96] for a survey) where a physics-informed GNN-based scalable general-purpose QUBO solver is proposed. The approach therein is suitable for encoding any  $k$ -local Ising model such as the  $k = 2$  ColOpt problem discussed later. The GNN solver first drops the integrality constraints in order to obtain a differentiable relaxation  $f'$  of the original objective function  $f$  and subsequently proceeds to unsupervised learning on the node representations. The GNN is then trained to generate soft assignments, pre-

dicting the likelihood of each vertex in the graph belonging to one of two distinct classes, in conjunction with heuristics that aid in the consistency of the problem. Interestingly, the authors benchmark this approach in demanding problem instances of MAXCUT to find that it competes with the best in class SDP algorithms such as the Goemans-Williamson algorithm [97].

**QUBO continuous relaxations with light sources.** Finally, let us mention a recent heuristic quantum-inspired (relaxation) approach for solving QUBOs, introduced in [98]. Concretely, the binary variables of the QUBO problem are represented by the relative phases of laser sources, transforming the discrete optimization problem into a continuous one. The lasers interact through a unique optical coupler, which uses programmable diffractive elements and additional optical components to control the interaction between all pairs of lasers, with a dynamic range of up to 8 bits. This design enables a fully connected network between all lasers, facilitating high-resolution pairwise interactions that are crucial in solving QUBO problems.

In [98] a benchmarking was performed for instances of the 3-regular 3-XORSAT problem and it was found that this method achieves significantly better TTS results as the instance size was increased. Despite this, seemingly incredible result, it is quite unclear whether this approach can perform on par for problems that do not have known poly-time solutions (for generic instances of 3-XORSAT there exists an algorithm with  $\mathcal{O}(N^{2.736\dots})$  complexity). In fact, it would be interesting to benchmark this approach against specialized SAT and MIP solvers.

**Simulated quantum annealing.** Obtaining quantum advantage for certain problem instances may be, in some cases, closer than one would anticipate by utilizing the technique of simulated quantum annealing (SQA) [99]. Specifically, in [99] a method to rigorously demonstrate that the Markov chain underlying SQA effectively samples the target distribution and discovers the *global minimum* of the spike cost function in *poly* time supporting is developed. While the analysis is limited to a very specific model and cannot be considered conclusive, the authors use interesting techniques such as initiating warm starts (a very popular technique applied in Deep Learning [100] as well as in QAOA [101]), from the adiabatic path and using the quantum ground state probability distribution to comprehend the stationary distribution of SQA.

Interestingly, SQA is considered effective, as compared to a classical algorithm, for optimization problems with spike (deep and narrow) global optima. As such, exploiting hybrid solvers that combine SQA and classical algorithms could offer an alternative approach [102].

### III. COLLATERAL OPTIMIZATION

In Section II, our main objective was to identify the most suitable formulations of the KnapsackProb as

a QUBO. However, for smaller-scale problems, MILP solvers are generally expected to perform better than heuristic, hybrid, and near-term quantum solvers. Therefore, in this section, we first formulate the Collateral Optimization problem ColOpt as a MILP and subsequently re-formulate it as a QUBO, similar to the approach taken in Section II.

The objective is to conduct several small-scale problems utilizing hybrid solvers with the purpose of algorithmic reformulation of the problem and its implementation. The ultimate goal is not to demonstrate the potential of hybrid or quantum alternatives for combinatorial optimization, but rather to establish an automated approach to solving the problem once hardware capabilities become more advanced.

#### A. ColOpt MILP Formulation

In order to mitigate the risk of a borrower defaulting on a loan, it is necessary for them to furnish collateral in the form of stocks, bonds, cash, or other assets to offset any outstanding exposure. In the present scenario, we consider a financial institution that has a collection (or inventory) of assets, indicated by  $\mathcal{I}$ , which must be allocated among a set of accounts, indicated by  $\mathcal{A}$ . We shall use the indices  $i$  and  $j$  to refer to an asset and an account, respectively. The total number of assets and accounts is given by  $n$  and  $m$ , respectively. It should be noted that our primary focus in this section is on the algorithmic reformulation of the problem and its implementation, which could be automated once sufficiently powerful hardware becomes available. Consequently, we shall run a set of small-scale problems employing hybrid solver emulators as in Sec. II. Our objective is not to demonstrate the superior performance of hybrid or quantum alternatives for combinatorial optimization but rather to heuristically identify the most appropriate formulation of the problem at hand.

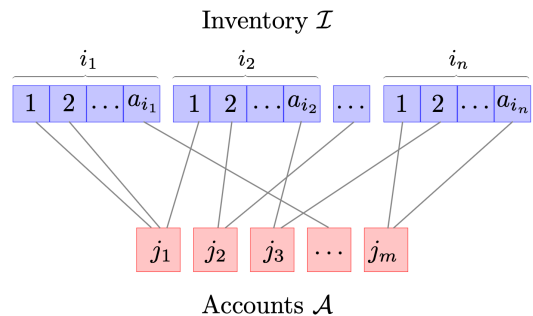


FIG. 1: Schematic representation of the collateral optimization problem. The figure above can be seen as a bipartite graph where one has to make optimal allocation of non-unique and weighted pairings.

Interestingly, ColOpt can be formulated as a bipartite matching problem as in Fig. 1. This bipartite graph is created with two sets of nodes: one set representing

the inventory of assets  $\mathcal{I}$ , and the other set representing accounts  $\mathcal{A}$ . The edges between these nodes represent potential allocations of assets to accounts, with weights on these edges representing the suitability, cost, or value of the allocation, and the edges are multi-directed and need to respect certain constraints. To model the constraints of ColOpt modifies the graph accordingly, as we will explain below.

**Limits.** Delving deeper into the problem, consider each asset  $i_k \in \mathcal{I}$ , where  $\mathcal{I} = i_1, i_2, \dots, i_n$ . Each asset  $i$  (momentarily simplifying the indices) is subdivided into a maximum quantity denoted by  $a_i$ . In more formal terms, there is a constraint on the maximum quantity of asset  $i$  that can be assigned. This is useful because, in the context of utilizing stocks as collateral, a financial institution may need or require the enforcement of an upper limit regarding the number of shares that can be allocated. The quantity of asset  $i_k$  can be converted into a corresponding dollar value by multiplying by the dimensionful term  $v_i$ , which is the market value (USD) per unit quantity.

**Tiers.** Every asset is linked to a tier, represented as  $\omega_i \in [0, 1]$ . This tier acts as a measure of the asset's quality, where distinct tiers correspond to various degrees of quality or attractiveness in the context of the ColOpt problem. The higher the value of  $\omega_i$ , the higher the quality of asset  $i$ .

**Exposure.** When a financial institution borrows from one of its lenders, collateral must be posted to adequately cover capital that could be lost in the event of a default. This capital requirement is known as "exposure". For each account  $j$ , there is a required exposure (in USD) that must be met indicated by  $c_j$ . Additionally, the duration of a transaction to a particular account can either be short term or long term. A binary variable  $d_j \in \{0, 1\}$  is used to indicate the duration of account  $j$ . A value 1 is assigned to short term and 0 for long term transfers. To reduce the risk of losing posted collateral, it is required to minimize the use of high quality assets for long term transactions whilst maximizing their use for short term transactions. We chose our decision variable to be a matrix  $Q \in \mathbb{Q}_{\leq 1}^{n \times m}$ , where the element  $Q_{ij}$  is the fractional amount of asset  $i$  that is allocated to account  $j$ . That is, the rows of  $Q$  correspond to the number of assets and the columns of the partition of each.

$$Q = \begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{nm} \end{pmatrix}. \quad (11)$$

To illustrate an objective function that represents our goals consider the simple case of allocating a single asset,  $i$ , across two accounts,  $j$  and  $l$ , which have long and short term requirements, respectively. In this case, the objective function can be formulated as:

$$\min \quad \omega_i Q_{ij} + (1 - \omega_i) Q_{il}. \quad (12)$$

In the expression above, the coefficient preceding short term allocations is set to  $1 - \omega_i$ . This is so that we favor allocations of higher quality assets for trades with a short duration. To generalize this for all assets and accounts, we need a mechanism which updates these tiers according to the type of account they are posting collateral towards. This can be done by constructing a coefficients matrix  $\Omega$ , where each element can be determined using:

$$\Omega_{ij} = |\omega_i - d_j|. \quad (13)$$

Just as with the tiers, each element of  $\Omega$  has a range of  $[0, 1]$ . The objective function is then just the sum of element-wise multiplications between  $\Omega$  and  $Q$ :

$$\min_Q \quad \sum_{i=1}^n \sum_{j=1}^m \Omega_{ij} Q_{ij}. \quad (14)$$

In order to post collateral such that the financial institution meets the exposure for each account, we include a requirement constraint:

$$\sum_{i=1}^n Q_{ij} a_i v_i H_{ij} \geq c_j \quad \forall j \in \mathcal{A}. \quad (15)$$

Here,  $v_i$  denotes the dollar market value for a single unit of quantity for asset  $i$ . Hence, the term on the left-hand side represents the dollar value for the quantity of collateral that is chosen to be allocated, adjusted by a fractional factor  $H_{ij}$  which is referred to as the haircut. Since markets are dynamic, the value of a posted collateral can diverge from its market value over time. In the case that the value drops below the required collateral value, the receiver is at risk. To avoid this, each account owner can evaluate the risk and place a haircut factor to reduce the value of an asset. The haircut is defined as the percentage difference between the market value and its value whilst used as collateral. For example, a haircut of 10% corresponds to  $H_{ij} = 0.9$ , meaning the collateral value is 90% of the original market value.

We need to ensure that we do not allocate more collateral than we have available in inventory (i.e. we do not allocate more than 100% of the maximum available quantity). In financial terms, this prevents us from short selling the asset. This is done by including a consistency constraint:

$$\sum_{j=1}^m Q_{ij} \leq 1 \quad \forall i \in \mathcal{I}. \quad (16)$$

There is also the trivial constraint to make sure that  $Q_{ij}$  does not take negative values:

$$Q_{ij} \geq 0 \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{A}. \quad (17)$$



**Further constraints that be can be imposed.** For example, there may be limits on the amount of a particular asset  $i$  to account  $j$ , given by  $B_{ij}$ . If  $B_{ij} = 0$ , the allocation is not eligible. This is a one-to-one constraint and has the following mathematical form:

$$Q_{ij}a_i \leq B_{ij} \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{A}. \quad (18)$$

Moreover, there are constraints that restrict the allocation of specific groups of assets to a single account, which exhibits a many-to-one relationship. For instance, certain types of assets, say  $\{i_{X_1}, i_{X_2}, i_{X_3}\}$ , may be subject to restrictions due to their interrelationships (for example there exists a parent company  $X$  that posts these assets). To formalize this constraint, we introduce  $\mathcal{G}$ , which represents the set of all groups of assets. The binary variable  $T_{ig}$  is used to indicate whether asset  $i$  belongs to the group  $g$ , while  $K_{gj}$  represents the upper bound on the total amount of assets from group  $g$  that can be allocated to account  $j$ :

$$\sum_{i=1}^n T_{ig} Q_{ij} a_i \leq K_{gj} \quad \forall g \in \mathcal{G}, \forall j \in \mathcal{A}. \quad (19)$$

The aim of imposing limits on the allocation of assets is to promote diversification and thereby reduce the risk borne by the receiver. In this paper, we concentrate on allocating cash rather than equity and bonds, which allows us to avoid the constraint that  $Q_{ij}a_i$  must take an integer value. Consequently, we can formulate the problem of collateral optimization as a continuous optimization problem without the need for additional constraints.

**The Complete ColOpt Problem.** Taking into account the information presented in the previous paragraphs, we can express the ColOpt problem using a MILP formulation:

$$\min \sum_{i=1}^n \sum_{j=1}^m \Omega_{ij} Q_{ij} \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{A}. \quad (20a)$$

$$\text{s.t.} \quad \sum_{j=1}^m Q_{ij} \leq 1 \quad \forall i \in \mathcal{I} \quad (20b)$$

$$Q_{ij}a_i \leq B_{ij} \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{A} \quad (20c)$$

$$\sum_{i=1}^n T_{ig} Q_{ij} a_i \leq K_{gj} \quad \forall g \in \mathcal{G}, \forall j \in \mathcal{A} \quad (20d)$$

$$\sum_{i=1}^n Q_{ij} a_i v_i H_{ij} \geq c_j \quad \forall j \in \mathcal{A} \quad (20e)$$

$$Q_{ij} \geq 0 \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{A}. \quad (20f)$$

For clarity, we will summarize the constraints presented above. Constraint (20b) ensures that no asset is distributed to the accounts beyond unity. Constraint (20c) amounts to the limit constraints for each asset-account pairing. Constraint (20d) limits the quantity of particular groups of assets to certain accounts but will be ignored

in what follows. Constraint (20e) is the requirement constraint that enforces that we allocate a suitable value such that the lender's loan is secured.

## B. ColOpt QUBO Formulation

**Binarization.** To formulate the QUBO, we need to make a change of variables so that our decision variable is represented by binaries. This change imposes certain limitations on the allocation of assets, which are discussed in detail below. We make use of an  $n$ -bit binary variable, similar to the methods used by Lang [103] and Ottaviani [104]. To enable binary encoding of the decision variable  $Q$ , we can represent it as a matrix  $q$  containing binary elements. This transformation enables us to only allocate assets in a limited number of ways, as detailed below:

$$q = \begin{pmatrix} \text{bin}_{11} & \dots & \text{bin}_{01} \\ \vdots & \ddots & \vdots \\ \text{bin}_{n1} & \dots & \text{bin}_{nm} \end{pmatrix}. \quad (21)$$

Here,  $\text{bin}_{ij}$  is an  $n$ -bit binary variable expressing the fractional allocation of asset  $i$  to account  $j$ :

$$\text{bin}_{ij} = x_{ij}^{b=1}, \dots, x_{ij}^{b=B}, \quad x_{ij}^b \in \{0, 1\}, \quad (22)$$

where  $B$  is the number of bits chosen.

Using  $B = 4$  as an example, the largest number which can be represented by four bits is  $1111_{\text{bin}} = 15_{\text{dec}}$ . Thus, we split our allocation into 15 fractions, where if  $\text{bin}_{ij} = 0100_{\text{bin}} = 4_{\text{dec}}$  then  $4/15$  of asset  $i$  is allocated to account  $j$ . By increasing  $B$ , we increase the precision of our allocations.

By implementing a similar method to Braun *et al.* [105], we can discretize our fractional allocation by discretizing the interval  $[Q_{ij}^{\min}, Q_{ij}^{\max}]$ :

$$Q_{ij}^{\max} - Q_{ij}^{\min} = \sum_{b=1}^B p_{ij}^b = \sum_{b=1}^B \frac{2^{(b-1)}(Q_{ij}^{\max} - Q_{ij}^{\min})}{M}. \quad (23)$$

$M$  is the maximum value which can be represented by a binary string of length  $B$  ( $M = 2^B - 1$ ). *Remark:*  $Q_{ij}^{\max}, Q_{ij}^{\min}$  have been included here for generality, but in our problem  $Q_{ij}^{\max} = 1$  and  $Q_{ij}^{\min} = 0$  by definition. Thus,  $Q_{ij}^{\max} - Q_{ij}^{\min}$  is always 1.

The discretized amount of item  $i$  that is allocated to account  $j$  is then described by the dot product of the binary vector  $x_{ijb} = (x_{ij1}, \dots, x_{ijB})^T$  and  $p_{ijb} = (p_{ij1}, \dots, p_{ijB})^T$ .

$$q_{ij} = \sum_{b=1}^B p_{ijb} x_{ijb} \equiv p_{ijb} x_{ijb}^b, \quad (24)$$

where we use the Einstein summation convention (upper-lower repeated indices contract) for brevity.

The cost function is then written as:

$$\sum_{i=1}^n \sum_{j=1}^m \Omega_{ij} p_{ijb} x_{ij}^b. \quad (25)$$

We then make the same replacement of  $Q_{ij} \rightarrow p_{ijb} x_{ij}^b$  in the remaining constraints of the problem to construct the binarized collateral optimization problem. Again, a wide number of classical solvers (open-source and commercial) are available which can find the global minimum of the problem in this form.

The total number of variables used to construct this binarized version of the problem is  $\mathcal{O}(nmB)$ . Since we have replaced the continuous variables with discrete binaries, the accuracy of the solution is expected to be reduced. This can be mitigated by increasing  $B$ , however, a compromise is needed between accuracy and resource usage. Regardless, the granularity for the fractional allocation is  $1/M$ .

**Slack-based Formulation.** We hereby introduce a QUBO formulation for the collateral optimization problem, incorporating slack variables. The constraints outlined in (20b) - (20e) significantly influence the number of bits necessary to encode these slack variables, potentially leading to an extensive bit requirement. To address this issue, we employ the log-encoding method earlier in Sec. II C.

For constraints expressed as “less-than-or-equal-to” inequalities, the number of bits required to encode the slack variable can be readily computed using  $\lceil \log_2 u \rceil$ , where  $u$  represents the upper bound of the constraint. Nonetheless, addressing the requirement constraint (20e) necessitates a more nuanced approach. As the objective is to minimize the excess value of the collateral posted, employing slack variables might prove inadequate, since their purpose is to diminish the corresponding penalty term to 0 for any values satisfying this constraint. In contrast, within MILP frameworks, the solution to a minimization problem typically aligns closely with the lower bound of such a constraint. In light of this observation, we choose to alter the exposure requirement by transforming it into an equality constraint, thereby relaxing the original formulation:

$$\sum_{i=1}^n Q_{ij} a_i v_i H_{ij} = c_j \quad \forall j \in \mathcal{A}, \quad (26)$$

and as a result, the associated penalty term requires no slack variables.

Hence, with the objective of minimizing the associated penalty terms originating from the aforementioned expression, the QUBO should yield a solution conforming to the boundaries of the exposure constraints. A limitation of this strategy is the intrinsic stochastic characteristic of annealing techniques and their propensity to become ensnared in local minima, potentially resulting in marginally exceeding or not quite meeting the mandated exposure. Furthermore, because the upper bound

for each consistency constraint is equal to one, we need to adjust their form so that we can calculate the number of slack bits required. We proceed by rearranging the binarized version of this constraint which then attains a fractional form:

$$\sum_{j=1}^m p_{ijb} x_{ij}^b = \sum_{j=1}^m \sum_{b=1}^B \frac{2^{b-1} x_{ij}^b}{M} \leq 1, \quad (27)$$

$\forall i \in \mathcal{I}$ . By multiplying both sides by  $M$ , it is straightforward to realize that the highest value that can be represented by the bitstring is the one we use as our new upper bound, and this further allows us to easily determine the number of slack bits needed.

The penalty term for each of the  $n$  consistency constraints is written as:

$$\sum_{i=1}^n \left( \sum_{j=1}^m M(p_{ijb} x_{ij}^b - 1) + S_{\text{con}} \right)^2, \quad (28)$$

where  $S_{\text{con}}$  is the slack variable for each constraint which is encoded by binary variables  $s_k$  via:

$$S_{\text{con}} = \sum_{k=1}^{\lceil \log_2(M) \rceil} 2^{k-1} s_k. \quad (29)$$

Instead of introducing a penalty term for one-to-one constraints (20c), we can ensure that these are satisfied by reducing the number of bits representing allocations so that they cannot violate these limits. The number of bits representing an allocation,  $n_{ij}$ , can be determined by:

$$n_{ij} = \left\lfloor \log_2 \left( \frac{B_{ij}}{a_{ij}} M \right) \right\rfloor. \quad (30)$$

We choose to floor the result from the logarithmic function, as the alternative would still allow violations. However, a consequence of this is that the one-to-one constraints in the QUBO are more restrictive than their MILP counterpart.

Aside from these nuances above and the additional step of binarization, constructing the balanced formulation for this QUBO follows the same process described above in the discussion of the `KnapsackProb` instance. For the many-to-one constraints (20d), we introduce log-encoded slack variables  $S_{K_{ij}}$ . It is straight-forward to derive the full QUBO objective as:

$$\begin{aligned}
& \lambda_0 \sum_{i=1}^n \sum_{j=1}^m \Omega_{ij} p_{ijb} x_{ij}^b a_i \\
& + \lambda_1 \sum_{i=1}^n \left( \sum_{j=1}^m M(p_{ijb} x_{ij}^b - 1) + S_{\text{con}} \right)^2 \\
& + \lambda_2 \sum_{j=1}^m \left( \sum_{i=1}^n p_{ijb} x_{ij}^b a_i v_i H_{ij} - c_j \right)^2 \\
& + \lambda_3 \sum_{j=1}^m \sum_{g=1}^G \left( \sum_{i=1}^n p_{ijb} x_{ij}^b T_{ig} a_i - K_{gj} + S_{K_{gj}} \right)^2.
\end{aligned} \tag{31}$$

**Unbalanced formulation.** The previous constraints, set in equations (20b) -(20f), can be converted into penalty terms for the QUBO through unbalanced penalization, as we displayed with the `KnapsackProb` instance in Sec. IID. For this approach, auxiliary functions are defined from the constraints, and Taylor approximations of appropriate exponentiations of these functions are used to derive the penalty terms.

For instance, consider the consistency constraint equation (20b). We move the upper bound to the LHS of the inequality and set this as our auxiliary function  $h(x)$ :

$$h(x) = \sum_{j=1}^m p_{ijb} x_{ij}^b - 1 \leq 0. \tag{32}$$

Since this is a ‘less-equal to zero’ inequality we use  $e^{h(x)}$  to derive a penalty term that takes small values when this constraint is satisfied and large values when violated. As mentioned previously, QUBOs may only contain linear and quadratic terms so, therefore, we need to take a 2<sup>nd</sup> order Taylor approximation to obtain:

$$\lambda_1 \left( \sum_{j=1}^m p_{ijb} x_{ij}^b - 1 \right) + \lambda_2 \left( \sum_{j=1}^m p_{ijb} x_{ij}^b - 1 \right)^2, \tag{33}$$

for all  $i \in \mathcal{I}$ . Essentially, the first of these terms favor solutions which satisfy the constraint whilst being as far away from the upper bound as possible. The second term, instead, favors solutions which are as close to this upper bound. Effective tuning of the parameters is therefore necessary to balance the effects of each term. Note, however, in this case we do not need to relax the exposure constraint to an equality as we can better manage how far beyond the solution is from these lower bounds. This equation is valid only for one asset, therefore we can modify equation (33) to consider all assets:

$$\begin{aligned}
& \lambda_1 \sum_{i=1}^n \left( \sum_{j=1}^m p_{ijb} x_{ij}^b - 1 \right) \\
& + \lambda_2 \sum_{i=1}^n \left( \sum_{j=1}^m p_{ijb} x_{ij}^b - 1 \right)^2.
\end{aligned} \tag{34}$$

Following the same methods and the discussion on the one-to-one constraints used in the previous formulation, we can promote the remaining constraints to penalty terms in the QUBO.

The final QUBO can be written as:

$$\begin{aligned}
& \lambda_0 \sum_{i=1}^n \sum_{j=1}^m \Omega_{ij} p_{ijb} x_{ij}^b \\
& + \lambda_1 \sum_{i=1}^n \left( \sum_{j=1}^m p_{ijb} x_{ij}^b - 1 \right) \\
& + \lambda_2 \sum_{i=1}^n \left( \sum_{j=1}^m p_{ijb} x_{ij}^b - 1 \right)^2 \\
& - \lambda_3 \sum_{j=1}^m \left( \sum_{i=1}^n p_{ijb} x_{ij}^b a_i v_i H_{ij} - c_j \right) \\
& + \lambda_4 \sum_{j=1}^m \left( \sum_{i=1}^n p_{ijb} x_{ij}^b a_i v_i H_{ij} - c_j \right)^2 \\
& + \lambda_5 \sum_{j=1}^m \sum_{g=1}^G \left( \sum_{i=1}^n p_{ijb} x_{ij}^b T_{ig} a_i - K_{gj} \right) \\
& + \lambda_6 \sum_{j=1}^m \sum_{g=1}^G \left( \sum_{i=1}^n p_{ijb} x_{ij}^b T_{ig} a_i - K_{gj} \right)^2
\end{aligned} \tag{35}$$

#### IV. NUMERICAL ILLUSTRATIONS

Utilizing the formulations presented in Sec. III we now define a small instance of the `ColOpt` problem based on a synthetic (however realistic) small dataset. We perform our tests on an Apple MacBook Pro with M2 Max processor and 16GB of memory. We remark that unlike the `KnapsackProb` which even with a small instance of a few items has a relatively simple structure with a single constraint, the small `ColOpt` problem instance we describe below has more complex constraints, and the interdependencies between the accounts and assets lead to a larger QUBO matrix and a much more challenging optimization landscape. In what follows, we utilize simulated annealing (SA), which as a metaheuristic algorithm, [106] is quite sensitive to the problem structure and its performance can vary significantly depending on the problem instance. The increased complexity in the collateral optimization problem may make it harder for the SA to explore the solution space effectively, leading to suboptimal results or longer convergence times. In such cases, it might be beneficial to fine-tune the parameters of SA to improve its performance on more complicated problem instances. An interesting approach towards that direction was followed in [33] which, for our context, we leave for future work.

Concretely, we have a portfolio of ten assets that have an approximate combined value of \$8.86M. These assets can be categorized by their tier rating,  $\omega = \{0.2, 0.5, 0.8\}$ ,

Formulation	Solver	Cost Function	Consistency				Exposure	Objective Value
		$\lambda_0$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_4$		
Balanced	D-Wave’s Simulated Annealing Sampler	$10^3$	1	-	1	-		0.5898
Balanced	Fujitsu’s Digital Annealer	$10^5$	1	-	300	-		0.7559
Unbalanced	D-Wave’s Simulated Annealing Sampler	$1.5 \times 10^4$	1	1	1	50		0.5244
Unbalanced	Fujitsu’s Digital Annealer	$2 \times 10^4$	1	1	1	50		0.5803
Continuous LP	HiGHS (Simplex)	-	-	-	-	-		0.4746

TABLE III: Values used for tuning the Lagrangian multipliers for each term in the QUBO for differing implementations. The objective value obtained for running this solution is also documented and for comparison, the value obtained by a continuous solver is displayed.

into low, mid, and high tiered assets, respectively. Furthermore, the number of assets belonging to each category is chosen to be 4, 2 and 4, respectively. These assets are to be distributed in order to meet the requirements of 5 accounts. These requirements are distinguished by their duration, two of which are long-term and have a combined exposure of  $\sim \$1.49\text{M}$ . The remaining are short-term requirements with a total exposure of  $\sim \$1.09\text{M}$ .

Due to restrictions, we slightly relax the problem by removing many-to-one constraints (20d). It is clear that in the absence of these constraints, this instance of the ColOpt has a global optimum that can be easily obtained using classical strategies. As mentioned earlier, a compromise was needed between the precision of our results and the run-time performance, along with the limitation of the total number of bits that can be implemented in the solvers. To do this, we set the length of the bitstring representing each allocation, (22), to be 7. Hence, the granularity of our allocations is  $1/127 \approx 0.0079\%$ , of the quantity of assets available ( $M = 127$ ). This becomes important if an asset quantity is significantly large, as the corresponding solution will allocate more than what is necessary to meet the requirements. Also, if very strict limits were considered, many violations could occur if the bitstring length chosen was inadequate. To this effect, we ensure that our sample contains sensible values for the quantity of each asset so that they can be distributed with enough precision to satisfy the exposure requirements efficiently.

Solving QUBO equations to obtain results that accurately reflect the goal of the objective function whilst simultaneously satisfying all constraints relies on the fine-tuning of the Lagrange multipliers. A potential solution is to make use of “hybrid solvers” such as D-Wave’s Constrained Quadratic Model (CQM) [107] which automatically calculates these, we instead use an intuitive approach. The consistency constraint (20b) is a hard constraint, as a solution that violates this constraint does not translate into a sensible business solution. Conversely, we treat the exposure requirement (20e) as a soft constraint and allow for small violations to some margin  $\epsilon$ , see Fig. 4. Additionally, for both balanced and unbalanced formulations, it is important to note that there are significant differences between the magnitudes of the coefficients of each term in the QUBO. This can make fine-tuning of the penalty weights a difficult task. To manage

this, we normalize each term in the QUBOs by dividing by their largest coefficient and then scale such that the lowest coefficient in each term has an order of magnitude of 1. We chose the weight for the cost function to be a magnitude larger than those of the constraints, so that we achieve high-quality solutions. We then retrospectively increase the weight of the exposure and consistency terms to ensure that the system’s constraints are satisfied.

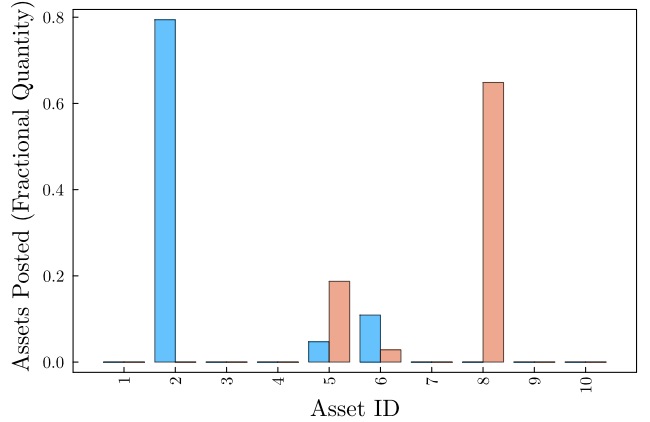


FIG. 2: The optimal allocations of assets among accounts with short term (red) and long term (blue) requirements, determined through solving the ColOpt instance as a continuous LP with HiGHS. Assets IDs of 1 – 4 are low-tier, 5 – 6 are mid-tier and the final 7 – 10 are the high-tier assets.

Overall, our results were mixed in the sense that none of our runs managed to reach the global optimum nor to produce the globally optimal allocation with each run converging in a different local minimum. We estimate that one reason for this behavior is the limited number of runs performed that do not allow the annealing process to explore sufficient search space. This can be easily redeemed by increasing the number of runs (potentially decreasing the step size) and utilizing more compute power.

Table III displays the values chosen for each of the penalty weights and the resultant objective value that was outputted. Fig. 2 shows the global optimal solution solved using HiGHS. The asset allocations using neal and Fujitsu’s Digital Annealer are shown in Fig. 3

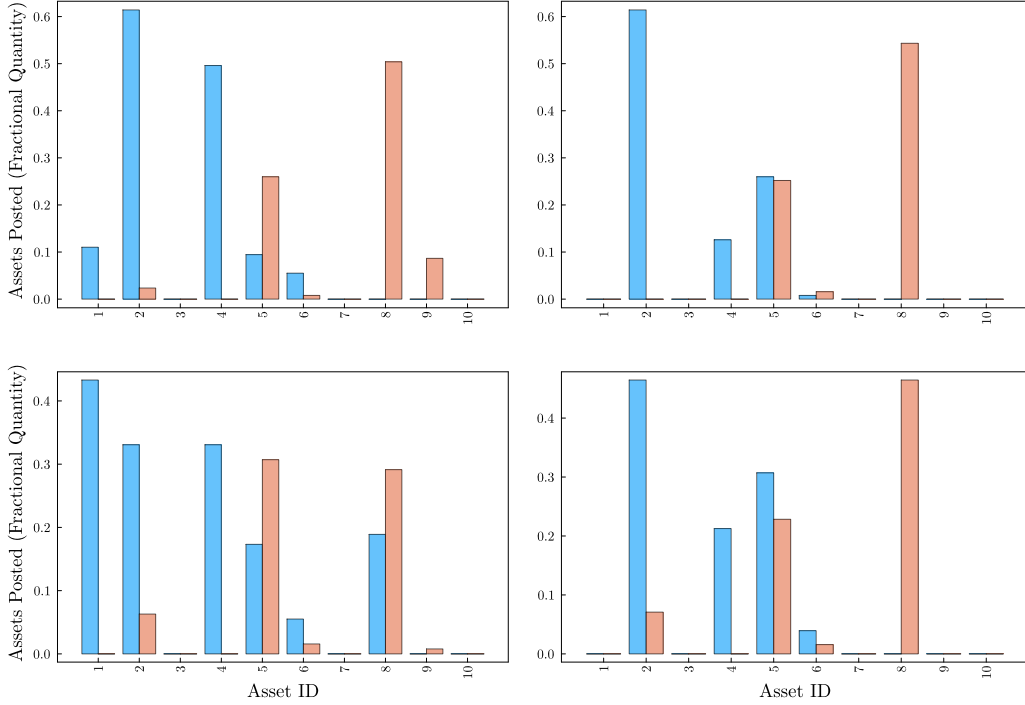


FIG. 3: The allocation, of different assets amongst accounts with short term (red) and long term (blue) requirements. Results are determined by D-Wave’s simulated annealer (top) and Fujitsu’s Digital annealer (bottom) with balanced (left) and unbalanced (right) formulations. The asset IDs of 1 – 4 are low-tier, 5 – 6 are mid-tier and the final 7 – 10 are the high-tier assets.

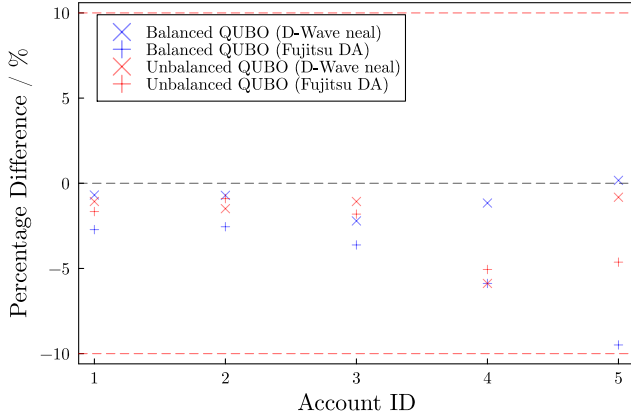


FIG. 4: The percentage of the exposure requirements that have been met for each account. The dashed line represents the solution given by HiGHS solver which perfectly meets each requirement. We see greater deviations for the requirement of account 4, which is due to its outstanding exposure being an order of magnitude less than those of the other accounts, hence it has a lower weighting in the QUBO.

## V. SUMMARY AND CONCLUSIONS

We surveyed the problem of collateral optimization from a business perspective and provided a business-realistic MILP formulation suitable to be mapped to a QUBO. In turn, we provided two QUBO formulations, one based on slack variables and one based on unbalanced penalization, and we implemented a small

ColOpt problem instance on small emulations of hybrid solvers. We observed that the unbalanced penalization approach yields objective function values much closer to the global optimum, obtained using the simplex method, while the slack-based (balanced) approaches were further off. To this end, both approaches fail to find the global optimum even for such a relatively small, but non-trivial, problem instance. While we did not run our computations on specialized hardware (quantum annealer or digital annealer), to some extent, we indeed rediscover the expectation that heuristic approaches fail to find globally optimal solutions very often, including when implemented in quantum hardware [108]. The “take-home message” from the numerical illustrations clearly showcases the relevance of the formulation chosen for a given problem when one is interested in making it quantum-ready.

Note, however, that the aim of this paper is to show the relevance as well as the technical formulation of the collateral optimization problem for quantum or hybrid solutions rather than to perform a detailed benchmarking and, as a matter of fact, several improvements can be performed in order to obtain higher quality solutions (warm starting techniques, optimizing the annealing schedule, QUBO parameter optimization, utilization of GPU and tensor cores, and explore improved methods for SA [109]).

The question of whether MILPs such as the collateral optimization problem should be cast as QUBOs and ap-

proached by heuristic solvers (quantum or not) is tricky. To some extent, classical solvers such as Gurobi, CPLEX, or even HiGHS perform exceptionally well in many large instances. Improving thereof could be less or more beneficial depending on the situation or problem instance and its complexity. If a “quantum” solution is preferred, finding the ultimate strategy for this approach is a crucial problem (formulation of the problem, hyperparameter choice, solver choice, etc).

The ColOpt problem, as presented above, can be extended in a variety of ways. The formulations provided above are in no way unique and further investigation might yield better results even for the local solvers. For example, recent research [110] has shown that an alternative formulation based on implicit penalization by restricting the Hamiltonian dynamics, in the context of parallelized QAOA, can be powerful and such an approach can be suitable for the collateral optimization problem as well. A different approach based on a stochastic quantum Monte Carlo algorithm, that mimics quan-

tum annealing, was proposed [111] for such large-scale optimization problems making it another suitable candidate for ColOpt. The benefit of this approach is that it can handle fully connected graphs which can be the case in certain ColOpt instances.

*Acknowledgements.* We would like to thank Jakub Marecek, Vyacheslav Kungurtsev, David Snelling, Reiss Pikett and Philip Intallura for useful discussions and suggestions.

*Disclaimer.* This paper was prepared for information purposes and is not a product of HSBC Bank Plc. or its affiliates. Neither HSBC Bank Plc. nor any of its affiliates make any explicit or implied representation or warranty and none of them accept any liability in connection with this paper, including, but limited to, the completeness, accuracy, reliability of information contained herein and the potential legal, compliance, tax or accounting effects thereof. Copyright HSBC Group 2023.

- 
- [1] M. Simmons, *Collateral management*, Wiley Finance (John Wiley & Sons, 2019).
  - [2] A. Nützenadel, *Journal of Modern European History* **19**, 3 (2020).
  - [3] *SIAM Review* (2017).
  - [4] P. M. McBride, *The International Lawyer* **44**, 1077 (2010).
  - [5] A. Delivorias, *European Parliamentary Research Service (EPRS)* (2017).
  - [6] D. Brigo, arXiv preprint arXiv:1111.1331 (2011).
  - [7] Deloitte, (2014).
  - [8] D. Bertsimas, *Introduction to linear optimization*, Optimization and Neural Computation Series (Athena Scientific, 1997).
  - [9] I. I. Cplex, *International Business Machines Corporation* **46**, 157 (2009).
  - [10] Gurobi Optimization, LLC, “Gurobi Optimizer Reference Manual,” (2023).
  - [11] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.* (2019).
  - [12] J. Bylund, “Collateral optimization,” (2017).
  - [13] S. M. Mniszewski, in *Proceedings of the International Conference on Neuromorphic Systems* (ACM, 2019).
  - [14] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, P. Date, and B. Kay, *Nature Computational Science* **2**, 10 (2022).
  - [15] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, and J. L. O’Brien, *nature* **464**, 45 (2010).
  - [16] D. J. Egger, C. Gambella, J. Marecek, S. McFaddin, M. Mevissen, R. Raymond, A. Simonetto, S. Woerner, and E. Yndurain, (2020), 2006.14510.
  - [17] D. Herman, C. Googin, X. Liu, A. Galda, I. Safro, Y. Sun, M. Pistoia, and Y. Alexeev, (2022), 2201.02773.
  - [18] D. J. Egger, C. Gambella, J. Marecek, S. McFaddin, M. Mevissen, R. Raymond, A. Simonetto, S. Woerner, and E. Yndurain, *IEEE Transactions on Quantum Engineering* **1**, 1 (2020).
  - [19] A. Bouland, W. van Dam, H. Joorati, I. Kerenidis, and A. Prakash, arXiv preprint arXiv:2011.06492 (2020).
  - [20] D. Herman, C. Googin, X. Liu, A. Galda, I. Safro, Y. Sun, M. Pistoia, and Y. Alexeev, arXiv preprint arXiv:2201.02773 (2022).
  - [21] P. Rebertrost, A. Luongo, S. Bosch, and S. Lloyd, arXiv preprint arXiv:2209.08867 (2022).
  - [22] P. Intallura, G. Korpas, S. Chakraborty, V. Kungurtsev, and J. Marecek, arXiv preprint arXiv:2303.04945 (2023).
  - [23] M. Pistoia, S. F. Ahmad, A. Ajagekar, A. Buts, S. Chakrabarti, D. Herman, S. Hu, A. Jena, P. Minssen, P. Niroula, *et al.*, in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)* (IEEE, 2021) pp. 1–9.
  - [24] A. Jacquier, O. Kondratyev, A. Lipton, and M. de Prado, *Quantum Machine Learning and Optimisation in Finance: On the Road to Quantum Advantage* (Packt Publishing, 2022).
  - [25] D. Emmanoulopoulos and S. Dimoska, arXiv e-prints, arXiv (2022).
  - [26] J. J. Kirk, M. D. Jackson, D. J. King, P. Intallura, and M. Metcalf, arXiv preprint arXiv:2303.01461 (2023).
  - [27] G. Rosenberg, P. Haghnegahdar, P. Goddard, P. Carr, K. Wu, and M. L. De Prado, in *Proceedings of the 8th Workshop on High Performance Computational Finance* (2015) pp. 1–7.
  - [28] P. Rebertrost and S. Lloyd, arXiv preprint arXiv:1811.03975 (2018).
  - [29] L. Leclerc, L. Ortiz-Gutierrez, S. Grijalva, B. Albrecht, J. R. Cline, V. E. Elfving, A. Signoles, L. Henriot, G. Del Bimbo, U. A. Sheikh, *et al.*, arXiv preprint arXiv:2212.03223 (2022).
  - [30] D. Lim and P. Rebertrost, arXiv preprint arXiv:2208.14749 (2022).
  - [31] S. Brandhofer, D. Braun, V. Dehn, G. Hellstern, M. Hüls, Y. Ji, I. Polian, A. S. Bhatia, and T. Wellens, *Quantum Information Processing* **22**, 1 (2023).



- [32] M. Vesely, arXiv preprint arXiv:2303.01909 (2023).
- [33] W. Sakuler, J. M. Oberreuter, R. Aiolfi, L. Asproni, B. Roman, and J. Schiefer, “A real world test of portfolio optimization with quantum annealing,” (2023), arXiv:2303.12601 [quant-ph].
- [34] A. Lucas, *Frontiers in physics* **2**, 5 (2014).
- [35] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, *Nature Reviews Physics* **3**, 625 (2021).
- [36] E. Farhi, J. Goldstone, and S. Gutmann, arXiv preprint arXiv:1411.4028 (2014).
- [37] Y. Chatterjee, E. Bourreau, and M. J. Rančić, arXiv preprint arXiv:2301.06978 (2023).
- [38] B. Apolloni, C. Carvalho, and D. de Falco, *Stochastic Processes and their Applications* **33**, 233 (1989).
- [39] A. Finnila, M. Gomez, C. Sebenik, C. Stenson, and J. Doll, *Chemical Physics Letters* **219**, 343 (1994).
- [40] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, *Science* **292**, 472 (2001).
- [41] T. Albash and D. A. Lidar, *Rev. Mod. Phys.* **90**, 015002 (2018).
- [42] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, *Reports on Progress in Physics* **83**, 054401 (2020).
- [43] D. Willsch, M. Willsch, C. D. Gonzalez Calaza, F. Jin, H. De Raedt, M. Svensson, and K. Michielsen, *Quantum Information Processing* **21**, 141 (2022).
- [44] F. Phillipson and H. S. Bhatia, in *Computational Science—ICCS 2021: 21st International Conference, Krakow, Poland, June 16–18, 2021, Proceedings, Part VI* (Springer, 2021) pp. 45–59.
- [45] P. J. M. Van Laarhoven and E. H. L. Aarts, *Simulated annealing: Theory and applications*, 1987th ed., Mathematics and Its Applications (Kluwer Academic, Dordrecht, Netherlands), (1987).
- [46] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. G. Katzgraber, *Frontiers in Physics* **7**, 48 (2019).
- [47] S. Tsukamoto, M. Takatsu, S. Matsubara, and H. Tamura, *FUJITSU Sci. Tech. J.* **53**, 8 (2017).
- [48] Z. Zhou, Y. Du, X. Tian, and D. Tao, *Phys. Rev. Appl.* **19**, 024027 (2023).
- [49] T. Lubinski, C. Coffrin, C. McGeoch, P. Sathe, J. Apanavicius, and D. E. B. Neira, arXiv preprint arXiv:2302.02278 (2023).
- [50] Ref. [51] defines a class NISQ, the class that contains all problems that can be solved by a polynomial-time probabilistic classical algorithm with access to a noisy quantum device and where  $BPP \subseteq NISQ \subseteq BQP$ .
- [51] S. Chen, J. Cotler, H.-Y. Huang, and J. Li, arXiv preprint arXiv:2210.07234 (2022).
- [52] L. Pusey-Nazzaro *et al.*, arXiv preprint arXiv:2008.07456 (2020).
- [53] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization : Algorithms and Complexity* (Dover Publications, 1998).
- [54] L. Huang, X. Chen, W. Huo, J. Wang, F. Zhang, B. Bai, and L. Shi, arXiv preprint arXiv:2111.06257 (2021).
- [55] G. Nannicini, *Phys. Rev. E* **99**, 013304 (2019).
- [56] L. Bittel and M. Kliesch, *Physical review letters* **127**, 120502 (2021).
- [57] J. Rivera-Dean, P. Huembeli, A. Acín, and J. Bowles, arXiv preprint arXiv:2104.02955 (2021).
- [58] V. Kungurtsev, G. Korpas, J. Marecek, and E. Y. Zhu, arXiv preprint arXiv:2209.10615 (2022).
- [59] G. De Palma, M. Marvian, C. Rouzé, and D. S. França, *PRX Quantum* **4**, 010309 (2023).
- [60] S. Ebadi, A. Keesling, M. Cain, T. T. Wang, H. Levine, D. Bluvstein, G. Semeghini, A. Omran, J.-G. Liu, R. Samajdar, *et al.*, *Science* **376**, 1209 (2022).
- [61] A. D. King, J. Raymond, T. Lanting, R. Harris, A. Zucca, F. Altomare, A. J. Berkley, K. Boothby, S. Ejtemaee, C. Enderud, E. Hoskinson, S. Huang, E. Ladizinsky, A. J. R. MacDonald, G. Marsden, R. Molavi, T. Oh, G. Poulin-Lamarre, M. Reis, C. Rich, Y. Sato, N. Tsai, M. Volkmann, J. D. Whittaker, J. Yao, A. W. Sandvik, and M. H. Amin, *Nature* (2023), 10.1038/s41586-023-05867-2.
- [62] M. Jünger, E. Lobe, P. Mutzel, G. Reinelt, F. Rendl, G. Rinaldi, and T. Stollenwerk, *ACM Journal of Experimental Algorithmics* **26**, 1 (2021).
- [63] F. Glover, G. Kochenberger, and Y. Du, arXiv preprint arXiv:1811.11538 (2018).
- [64] A. Montanez-Barrera, A. Maldonado-Romo, D. Willsch, and K. Michielsen, arXiv preprint arXiv:2211.13914 (2022).
- [65] K. Axiotis and C. Tzamos, arXiv preprint arXiv:1802.06440 (2018).
- [66] M. Bateni, M. Hajiaghayi, S. Seddighin, and C. Stein, in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing* (2018) pp. 1269–1282.
- [67] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations* (Wiley, 1990).
- [68] Q. Huangfu and J. A. J. Hall, *Mathematical Programming Computation* **10**, 119 (2017).
- [69] “Gnu linear programming kit,” <https://www.gnu.org/software/glpk/>.
- [70] P. Xavier, T. Andrade, J. Garcia, and D. Bernal, “ToQUBO.jl,” (2022).
- [71] Q. O. D. Team, “Quadraticprogramtoqubo,” .
- [72] M. Zaman, K. Tanahashi, and S. Tanaka, *IEEE Transactions on Computers* **71**, 838 (2021).
- [73] A. Glos, A. Kundu, and Ö. Salehi, arXiv preprint arXiv:2203.15421 (2022).
- [74] S. Palmer, S. Sahin, R. Hernandez, S. Mugel, and R. Orus, arXiv preprint arXiv:2106.06735 (2021).
- [75] L. Braine, D. J. Egger, J. Glick, and S. Woerner, *IEEE Transactions on Quantum Engineering* **2**, 1 (2021).
- [76] L. Braine, D. J. Egger, J. Glick, and S. Woerner, *IEEE Transactions on Quantum Engineering* **2**, 1 (2021).
- [77] K. Domino, A. Kundu, Özlem Salehi, and K. Krawiec, *Quantum Information Processing* **21** (2022), 10.1007/s11128-022-03670-y.
- [78] D. Venturelli, D. J. Marchand, and G. Rojo, arXiv preprint arXiv:1506.08479 (2015).
- [79] J. Zhang, G. L. Bianco, and J. C. Beck, *Proceedings of the International Conference on Automated Planning and Scheduling* **32**, 404 (2022).
- [80] D. Amaro, M. Rosenkranz, N. Fitzpatrick, K. Hirano, and M. Fiorentini, *EPJ Quantum Technology* **9** (2022), 10.1140/epjqt/s40507-022-00123-4.
- [81] K. Domino, M. Koniorczyk, K. Krawiec, K. Jałowiecki, S. Deffner, and B. Gardas, *Entropy* **25**, 191 (2023).
- [82] Y. Matsumoto and S. Nakamura, *SSRN Electronic Journal* (2022), 10.2139/ssrn.4037935.
- [83] A. Luckow, J. Klepsch, and J. Pichlmeier, *Digitale Welt* **5**, 38 (2021).

- [84] D. Ratke, blog.xa0.de (2021).
- [85] G. Evenbly and G. Vidal, *Journal of Statistical Physics* **145**, 891 (2011).
- [86] J. C. Bridgeman and C. T. Chubb, *Journal of physics A: Mathematical and theoretical* **50**, 223001 (2017).
- [87] D. A. Zheltkov and A. Osinsky, in *Large-Scale Scientific Computing* (Springer International Publishing, 2020) pp. 197–202.
- [88] A. Nikitin, A. Chertkov, R. Ballester-Ripoll, I. Osledeets, and E. Frolov, arXiv preprint arXiv:2205.04490 (2022).
- [89] B. C. B. Symons, D. Galvin, E. Sahin, V. Alexandrov, and S. Mensa, “A practitioner’s guide to quantum algorithms for optimisation problems,” (2023), arXiv:2305.07323 [quant-ph].
- [90] R. A. Quintero and L. F. Zuluaga, .
- [91] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods* (Elsevier, 1982).
- [92] I. Dunning, J. Huchette, and M. Lubin, *SIAM Review* **59**, 295 (2017).
- [93] A. Yurtsever, T. Birdal, and V. Golyanik, in *Computer Vision—ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIII* (Springer, 2022) pp. 352–369.
- [94] A. Gilliam, S. Woerner, and C. Goniculea, *Quantum* **5**, 428 (2021).
- [95] M. J. A. Schuetz, J. K. Brubaker, and H. G. Katzgraber, arXiv preprint arXiv:2107.01188 (2021).
- [96] P. Veličković, *Current Opinion in Structural Biology* **79**, 102538 (2023).
- [97] M. X. Goemans and D. P. Williamson, *Journal of the ACM* **42**, 1115 (1995).
- [98] I. Meirzada, A. Kalinski, D. Furman, T. Armon, T. Vaknin, H. Primack, C. Tradonsky, and R. Ben-Shlomi, arXiv preprint arXiv:2207.09517 (2022).
- [99] E. Crosson and A. W. Harrow, in *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE, 2016) pp. 714–723.
- [100] I. Loshchilov and F. Hutter, arXiv preprint arXiv:1608.03983 (2016).
- [101] D. J. Egger, J. Mareček, and S. Woerner, *Quantum* **5**, 479 (2021).
- [102] D. Volpe, G. A. Cirillo, M. Zamboni, and G. Turvani, *IEEE Access* **11**, 30390 (2023).
- [103] J. Lang, S. Zielinski, and S. Feld, *Applied Sciences* **12**, 12288 (2022).
- [104] D. Ottaviani and A. Amendola, arXiv preprint arXiv:1808.08721 (2018).
- [105] M. Braun, T. Decker, N. Hegemann, S. Kerstan, and F. Lorenz, arXiv preprint arXiv:2301.01108 (2023).
- [106] J. Mareček, *The Computer Journal* **53**, 1338 (2010).
- [107] D-Wave, “Constrained quadratic models,” .
- [108] D. Vert, R. Sirdey, and S. Louise, *SN Computer Science* **2**, 1 (2021).
- [109] H. Nakano and K. Fujiyoshi, *space* **1**, 2 (2016).
- [110] K. Ender, A. Messinger, M. Fellner, C. Dłaska, and W. Lechner, *PRX Quantum* **3**, 030304 (2022).
- [111] N. Onizawa, R. Sasaki, D. Shin, W. J. Gross, and T. Hanyu, “Stochastic quantum monte carlo algorithm for large-scale combinatorial optimization problems,” (2023).
- [112] B. A. Cipra, *The American Mathematical Monthly* **94**, 937 (1987).
- [113] However, the sign of  $J_{jk}$  does have a more interest-

ing interpretation. If  $J_{jk} > 0$  the model describes ferromagnetism, while if  $J_{jk} < 0$  it describes anti-ferromagnetism.

## Appendix A: Collateral optimization terminology

Below, in Table IV, we collect a few of the common terms used in the context of ColOpt and, more generally, collateral management.

Term	Definition
<b>Collateral</b>	Any valuable asset used within lending agreements which can be seized by the lender from the borrower if they fail to repay the loan. This can consist of any asset deemed valuable by the lender. Acceptable types of assets are: equity (stocks), bonds and cash.
<b>Market Value</b>	The price that an asset could be sold for when auctioned in the marketplace.
<b>Haircut</b>	A reduction applied to the market value of an asset. Used to ensure that the posted asset will cover the cost of the outstanding exposure, given the risk that an assets value can fluctuate.
<b>Exposure</b>	The amount of capital that one stands to lose (risk) if an investment fails.

TABLE IV: Definition of financial terms used.

Table V collects all relevant ColOpt related quantities/variables used in the main body of this paper.

Symbol	Explanation
$\mathcal{I}$	Set of $n$ inventory assets.
$i$	Index representing a particular asset.
$a_i$	Maximum available quantity of asset $i$ .
$v_i$	Dollar value of a single unit of asset $i$ .
$\omega_i$	Tier value representing the quality of asset $i$ .
$\mathcal{A}$	Set of $m$ accounts.
$j$	Index representing a particular account.
$c_j$	Dollar value exposure of account $j$ required.
$d_j$	Binary value indicating whether account obligation $j$ is short or long term.
$\Omega_{ij}$	Tier value of asset $i$ depending duration of account $j$ .
$Q_{ij}$	Decision variable indicating the percentage of asset $i$ allocated to account $j$ .
$B_{ij}$	Limit on individual allocation of asset $i$ to account $j$ .
$H_{ij}$	Haircut factor used to reduce the value of an asset.
$K_{gj}$	Limit on the allocation of assets in group $g$ to account $j$ .
$T_{ig}$	Binary value indicating whether asset $i$ belongs to group $g$ .

TABLE V: List of all symbols used in the formulation of the collateral optimization problem. The explanations are further described in Appendix A.



## Appendix B: Constraint Penalization in QUBOs and the Ising model

In the main body of this paper, we have seen that in order to encode penalty terms into QUBOs we need to use, for example, slack variables. Here we summarize how we proceed to do so. In the `KnapsackProb` as well as the collateral optimization problem, we have constraints of the form

$$Ax \leq c \quad (\text{B1})$$

$$Bx \geq d. \quad (\text{B2})$$

Let us consider the former constraint since the latter is essentially the same up to a factor of  $-1$ . It is common to view this constraint as  $Ax - c \leq 0$  and then introduce a slack variable  $s \in \mathbb{R}_{\geq 0}$  such that we define the penalty term  $Ax - c + s = 0$ . This is defined such that

$$\sup s = c - \inf Ax. \quad (\text{B3})$$

Then, as mentioned already in Sec. II, one needs to map a QUBO to the Ising Hamiltonian that performs Quantum Annealing (QA), a restricted model of adiabatic quantum computation. Let us now briefly explain this connection.

The (classical) Ising model was first introduced as a mathematical model of ferromagnetism [112]. The variables take values in a discrete (binary) set  $\sigma_i = \{\pm 1\}$ , and are typically referred to as *spin*, since in the physical model they describe the atomic spin of the particles. The model consists of a lattice,  $\Lambda$ , with each lattice site,  $j \in \Lambda$ , having an assigned spin  $\sigma_j$ . The energy of a specific spin configuration is measured by the Hamiltonian of the system

$$H = - \sum_{j < k} J_{j,j+1} \sigma_j \sigma_{j+1} - \sum_j h_j \sigma_j + \varepsilon,$$

where  $J_{j,j+1}$  encodes the “nearest neighbour interaction” between adjacent sites, and  $h_j$  describes some external field that interacts with each site [34]. The overall minus sign is just a convention [113] and  $\varepsilon$  refers to a constant energy overhead.

In the quantum version of the Ising Hamiltonian, the sites are represented by a qubit. The spins are then simply given by the Pauli matrix  $\sigma_j^z$  acting on the  $j$ ’th site, with eigenvalues  $\pm 1$  when acting on the computational basis states  $\{|0\rangle, |1\rangle\}$ . In physics, the interesting problem is typically to find the ground state energy, or lowest energy eigenstate, of the Hamiltonian.

One thing we can immediately notice is that this model is quadratic in the spins and thus resembles the QUBO. By the simple change of variables  $x_j = \frac{1-\sigma_j}{2}$  we directly see that the spin variables  $\pm 1$  are mapped to the binary variables  $\{0, 1\}$ . In the quantum version we map the QUBO variables  $x_j$  to the operators  $x_j = \mathbf{1} \otimes \dots \otimes \frac{1}{2}(\mathbf{1} - \sigma_z) \otimes \mathbf{1} \otimes \dots$ , where the non-trivial operator acts on the  $j$ ’th site. This operator has eigenvalues 0 or 1 when acting on the states  $|0\rangle$  respectively  $|1\rangle$ . Through this change of variables, the QUBO problem is thus equivalent to finding the ground state energy of the Ising Hamiltonian. When mapping between the QUBO and the Ising model we might also need to account for the fact that we only minimise over the Ising model, while sometimes, for example in the `KnapsackProb`, we are seeking a maximum. This is of course easily accounted for by changing the relevant signs.

As an example, the `KnapsackProb` (7) has an Ising Hamiltonian with:

$$\begin{aligned} J_{j,j+1} &= -\frac{\lambda_0}{4} w_j w_{j+1}, \\ h_j &= -\left(\frac{v_i}{2} + \lambda_0 K w_i\right), \\ c &= \frac{\lambda_0}{4} \left(\sum_i^n w_i\right)^2 - \frac{1}{2} \sum_i^n v_i - \lambda_0 K, \end{aligned}$$

where we define  $K = S - W + \frac{1}{2} \sum_i^n w_i$  for convenience.