Quantum Science and Technology

PAPER • **OPEN ACCESS**

Optimized QUBO formulation methods for quantum computing

To cite this article: Dario De Santis *et al* 2026 *Quantum Sci. Technol.* **11** 015056

IOP Publishing Quantum Sci. Technol. 11 (2026) 015056 https://doi.org/10.1088/2058-9565/ae3b70

Quantum Science and Technology

PAPER

OPEN ACCESS

RECEIVED
18 March 2025

REVISED
4 January 2026

ACCEPTED FOR PUBLICATION
21 January 2026

PUBLISHED
5 February 2026

Optimized QUBO formulation methods for quantum computing

Dario De Santis[1,*], **Salvatore Tirone**[1,2,3], **Stefano Marmi**[1] and **Vittorio Giovannetti**[4]

[1] Scuola Normale Superiore, Piazza dei Cavalieri 7, I-56126 Pisa, Italy
[2] QuSoft, Science Park 123, 1098 XG Amsterdam, The Netherlands
[3] Korteweg–de Vries Institute for Mathematics, University of Amsterdam, Science Park 105-107, 1098 XG Amsterdam, The Netherlands
[4] NEST, Scuola Normale Superiore and Istituto Nanoscienze-CNR, I-56126 Pisa, Italy
[*] Author to whom any correspondence should be addressed.

E-mail: dario.desantis@sns.it

Keywords: QUBO, computing, annealing, algorithm, heuristics, finance, optimization

Abstract

Quantum computers have strict requirements for the problems that they can efficiently solve. One of the principal limiting factor for the performances of noisy intermediate-scale quantum (NISQ) devices is the number of qubits required by the running algorithm. Several combinatorial optimization problems can be solved with NISQ devices once that a corresponding quadratic unconstrained binary optimization (QUBO) form is derived. Numerous techniques have been proposed to achieve such reformulations and, depending on the method chosen, the number of binary variables required, and therefore of qubits, can vary considerably. The aim of this work is to drastically reduce the variables needed for these QUBO reformulations in order to unlock the possibility to efficiently obtain optimal solutions for a class of optimization problems with NISQ devices. This goal is achieved by introducing novel tools that allow an efficient use of slack variables, even for problems with non-linear constraints, without the need to approximate the starting problem. We divide our new techniques in two independent parts, called the iterative quadratic polynomial and the master-satellite methods. Hence, we show how to apply our techniques in case of an NP-hard optimization problem inspired by a real-world financial scenario called Max-Profit Balance Settlement. We follow by submitting several instances of this problem to two D-wave quantum annealers, comparing the performances of our novel approach with the standard methods used in these scenarios. Moreover, this study allows to appreciate several performance differences between the D-wave Advantage and next-generation Advantage2 quantum annealers. We show that the adoption of our techniques in this context allows to obtain QUBO formulations with significantly fewer slack variables, i.e. around 90% less, and D-wave annealers provide considerably higher correct solution rates, which moreover do not decrease with the input size as fast as when adopting standard techniques.

1. Introduction

In recent years, many efforts have been devoted to understand to what extent modern noisy intermediate-scale quantum (NISQ) devices can help to solve optimization problems of any sort (see [1–3] for recent reviews). Particular attention has been paid to their potential to solve NP-hard and NP-complete problems, which most of the times can be formulated as quadratic unconstrained binary optimizations (QUBO). For instance, all famous NP Karp's problems [4] can be casted in this form [5].

The possibility to solve QUBO problems with quantum devices has been studied on several supports, especially on quantum annealers [1–3, 6–8], which are natively engineered as QUBO solvers and are particularly efficient to solve complex optimization problems compared to other NISQ devices, but also on logical-gate quantum computers [9–12] and Rydberg atom arrays [13–15]. Although these technologies are not expected to solve NP problems efficiently [16, 17], namely providing exponential speed-ups compared to classical strategies, possible polynomial in-time advantages attract great interest. Indeed, given

the importance of the real-world applications involved with the solutions of these hard combinatorial problems [2], any achievable improvement is well-received. Exemplary scenarios come from finance [18–23], logistics [24–26] and drug discovery [27].

Consider the situation where we aim to solve a generic constrained optimization problem with a QUBO solver. The first step is to reformulate it in a quadratic unconstrained form, namely as a QUBO. We call *logical variables* those defining the initial problem. The constraints attached to the original problem can be enforced in a QUBO form by employing additional *slack variables*. The larger is the total number of variables, logical and slack, in a QUBO problem, the harder is in general to solve it.

Whereas the well-established procedures to translate optimization problems into QUBOs [28] can be efficient in several scenarios, their implementation can be extremely inefficient at times. To be more precise, we say that a method to obtain these formulations is inefficient for a problem whenever it requires too many slack variables, namely their number is comparable with that of logical variables. In these cases, the practical usefulness of QUBO solvers can be highly limited. For instance, if we have a maximum size for the problems that our solver can receive, any reduction of slack variables allows to increase the total number of logical variables associated to the initial problem, namely larger problems can be tackled. Moreover, the more slack variables we implement, the higher is the connectivity that we require in our QUBO and therefore in the QUBO solver that we aim to use. This consequence can be a very limiting factor for the performances of modern NISQ devices.

The main motivation of this work is to unlock the possibility to solve certain classes of optimization problems with those quantum technologies specialized to solve QUBO problems. For this purpose, we introduce the iterative quadratic polynomial (IQP) and master-satellite (MS) methods. Their goal is to provide QUBO forms requiring a minimal employment of slack variables. The former consists in a new paradigm to translate problem constraints into corresponding quadratic penalty forms, while the latter allows the simultaneous enforcement of different constraints sharing the same restricted set of binary variables. Noteworthy, our techniques can also be used to enforce non-linear equality and inequality constraints defined through integer or non-integer parameters, without the need to approximate the constraints or to employ slack variables solely to obtain corresponding linearizations. Hence, the same computational effort is required both for linear and non-linear constraints, where no approximate enforcing of the constraints is required. Moreover, the QUBO variables employed by our methods for each constraint are not necessarily completely connected. These features allow improving the output quality of NISQ devices used as QUBO solvers as fewer and less-connected qubits are in general required.

We apply these techniques in the context of an NP-hard problem coming from finance, namely the Max-Profit Balance Settlement (MPBS) problem [29]. We provide a drastic reduction of the slack variables necessary to enforce the corresponding inequality constraints. We generate several instances of this problem reflecting some main features of realistic datasets [30] and we show that our methods provide the corresponding QUBO forms by employing around 90% less slack variables, if compared with the QUBO forms obtainable with a standard procedure [28].

We solve several instances of MPBS with two different quantum annealers manufactured by D-Wave Systems, Inc. [31], namely the D-wave Advantage_system4.1 and Advantage2_prototype1.1. The annealer Advantage2_prototype1.1, being a prototype of the next-generation D-wave annealers, has a reduced number of qubits but is characterized by higher connectivity and lower noise. Hence, not only do we compare the outputs obtained when the QUBO problems are generated with different approaches, but we also provide a comparison between the performances of these annealers. We show that, while by using the standard approach the solution quality drops quite fast with the input size, our methods unlock the possibility to consider much larger instances. The number of successes that we obtain when the QUBO forms are generated with our methods are always significantly higher: the multiplicative factor between the successes obtainable with the two methods ranges from 7 (smallest instances with Advantage_system4.1) to 184 (largest instances with Advantage2_prototype1.1).

We point out the existence of other techniques attempting to reduce the slack variables employed for QUBO reformulations. For instance, the alternating direction method of multipliers [32] and the augmented Lagrangian method [33, 34] have recently been adopted in this context [35–37], where preliminary QUBO problems have to be solved in order to get the final QUBO formulation of the original optimization problem. Another approach consists in enforcing constraints in an approximate fashion via a finely-tuned rescaling of the parameters defining the target linear inequalities [38–40]. Differently from these strategies, neither we require to solve preliminary QUBOs nor we approximate the starting optimization problem. Rather, we change the paradigm under which constraints are enforced.

The content of this work is structured as follows. In section 2, we introduce the fundamental tools needed throughout this work, as the definition of graph-based problems, QUBO forms and the standard

technique to achieve these quadratic forms. The first part of section 3 is devoted to describe in an intuitive way the main ideas behind of our novel methods. Instead, the IQP and the MS methods are respectively described in details in sections 3.1 and 3.2. The latter method requires a tuning of the QUBO penalty multipliers that is radically different from those found in standard QUBO formulations. Different strategies for this tuning are presented in section 3.3. The last heuristic that we introduce is a generalization of the MS approach, suitable for those cases where three or more constraints are defined over the same variables, which is proposed in section 3.4. We identify a class of optimization problems, namely the locally-constrained problems, in section 4, which are particularly suitable for the application of our techniques. The definition of the NP-hard MPBS problem is given in section 5, where we also analyse its QUBO formulation and the corresponding needs in terms of slack variables when using standard techniques. Instead, we show how to apply our novel techniques to MPBS step-by-step in section 6. The corresponding improved performance obtainable in the context of quantum annealing is shown in section 7, where the novel and the standard strategies are put in comparison.

2. Main tools

QUBO problems are defined as optimizations of quadratic forms defined over a set of binary variables $\mathbf{x} = \{0,1\}^N$, without the possibility to attach any constraint [28]. Hence, we have the following equivalent notations:

$$\min_{\mathbf{x}}/\max_{\mathbf{x}} \mathbf{x}^{\mathrm{T}} Q \mathbf{x} = \min_{\mathbf{x}}/\max_{\mathbf{x}} \sum_{i,j=1}^{N} Q_{ij} x_i x_j = \min_{\mathbf{x}}/\max_{\mathbf{x}} Q(\mathbf{x}) \,, \tag{1}$$

where Q is an upper-triangular $N \times N$ real-valued matrix and the corresponding optimization can either be a minimization or a maximization. Although these two cases are computationally equivalent, we consider the maximization case from now on.

Quantum annealers are the NISQ QUBO solvers that we adopt later in this work to solve these optimization problems, which in general are NP-hard [5]. The explanation of how quantum annealers work in order to obtain optimal solutions of QUBO problems can be found, e.g. in [1–3]. Considering the classical computation framework, optimization problems are tackled with several possible heuristics. An example of such techniques is called simulated annealing [41–43], the classical counterpart of quantum annealing, which searches into the space of possible solutions, while slowly decreasing the probability to accept worse alternatives of the current solution. A comparison between quantum and simulated annealing can be found in [44]. A different heuristic designed for the solution of QUBO problems is called tabu search, which is put in contrast with simulated annealing in [45]. At the end of this work, after analyzing how our strategies improve the performances of quantum annealers in solving optimization problems, we propose a similar comparison in the context of simulated annealing.

We proceed by explaining how several optimizations can be translated into a QUBO problem and where does the necessity to introduce new methods for this translation come from. Consider an optimization problem defined through the maximization of an objective function, which from now on we call W, where some constraints on the possible solutions must be satisfied. Usually, from the very definition of the objective function, we obtain a natural set of logical variables $\mathbf{x} = (x_1, x_2, \dots)$ associated to the units that describe the problem itself. For instance, if the problem requires to find a particular subgraph of an input graph, the binary variables may be associated to its arcs and/or nodes, or, if we aim to subdivide a particular quantity among different parties, the binary variables may be used to describe the corresponding fractions. Whenever $W(\mathbf{x})$ can be written as a linear or quadratic form of the variables $(x_1, x_2, x_3, \dots)$, its QUBO form is obtained straightforwardly. Nonetheless, in order to translate the constraints attached to our optimization problem in a QUBO form, we usually need additional binary variables $\mathbf{s} = (s_1, s_2, \dots)$, called slack variables, which are employed solely to enforce constraints [28].

This work is devoted to expose novel techniques to implement constraints in QUBOs. We point out that, even if some constraints could be particularly demanding in terms of slack variables, others may be not. Hence, whenever we use the expression 'target constraints', we refer to those on which we decide to apply our novel tools.

We study the generic situation of problems defined on directed, weighted and node-attributed multigraphs, namely:

Definition 1. (R-multigraph) *An R-multigraph consists of the quadruple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w, \mathbf{f})$, where $\mathcal{V}$ is a set of nodes, $\mathcal{E}$ is a multiset of ordered pairs of nodes called arcs, $w : \mathcal{E} \to \mathbb{R}^+$ is a function assigning positive real numbers, called weights, to arcs and $\mathbf{f} : \mathcal{V} \to \mathbb{R}^n$ is a function assigning real-valued n-dimensional vectors, called attributes, to nodes.*

Hence, we allow multiple arcs $(u,v) \in \mathcal{E}$ between two nodes $u,v \in \mathcal{V}$, where each arc $(u,v) \in \mathcal{E}$ has an origin u, a target v node and a positive weight $w(u,v)$. Moreover, each node $u \in \mathcal{V}$ may have attributes given by a real-valued n-dimensional vector $\mathbf{f}(u) = (f_1(u), \ldots, f_n(u))$. Naturally, what follows can be applied to simpler graphs, e.g. undirected and/or unweighted.

The majority of the optimization problems considered in the literature can be casted as follows: find the sub-graph $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*, w, \mathbf{f})$ of a given R-multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w, \mathbf{f})$ that maximizes the quantity $W(\hat{\mathcal{G}})$ under some given constraints, where $\hat{\mathcal{G}}$ is a generic sub-graph of $\mathcal{G}$. In order to write these problems in a more compact form, we introduce a binary variable $x_i = \{0,1\}$ for each arc $(u,v) \in \mathcal{E}$. Hence, we obtain a bijection $(u,v) \leftrightarrow x_i$, where $i = 1,2,\ldots,N$ and $N = |\mathcal{E}|$. Each sub-graph $\hat{\mathcal{E}} \subseteq \mathcal{E}$ is identified by a single combination of $\mathbf{x} = \{x_1, x_2, \ldots, x_N\} \in \{0,1\}^N$, where $x_i = 1$ if the corresponding arc is included in $\hat{\mathcal{E}}$ and $x_i = 0$ otherwise. Given this bijection, each combination of $\mathbf{x}$ defines a different $\hat{\mathcal{E}} \subseteq \mathcal{E}$ and $\hat{\mathcal{V}}$, where $u \in \hat{\mathcal{V}}$ if there exists at least one arc in $\hat{\mathcal{E}}$ such that u is either the origin or the target node. For instance, $\mathcal{E} \leftrightarrow \mathbf{x} = \{1,1,\ldots,1\}$.

We start by considering the class of optimization problems that have objective functions and constraints that are, respectively, (at most) quadratic and linear in the binary variables x_i. In case of $m_{\mathrm{EC}} \geqslant 0$ equality constraints $\mathrm{EC}_i(\mathbf{x}, w, \mathbf{f}) = 0$ and $m_{\mathrm{IC}} \geqslant 0$ inequality constraints $\mathrm{IC}_j(\mathbf{x}, w, \mathbf{f}) \leqslant 0$, an optimization problem of this kind can be written as:

$$\begin{aligned} \mathcal{E}^* = \arg \quad & \max_{\mathbf{x}} \quad W(\mathbf{x}, w, \mathbf{f}) \\ & \text{s.t.} \quad \mathrm{EC}_i(\mathbf{x}, w, \mathbf{f}) = 0 \quad \text{for } i = 1, \ldots, m_{\mathrm{EC}} \\ & \qquad \mathrm{IC}_j(\mathbf{x}, w, \mathbf{f}) \leqslant 0 \quad \text{for } j = 1, \ldots, m_{\mathrm{IC}} \end{aligned} . \tag{2}$$

Moreover, we assume the parameters defining the linear inequality constraints to be integer (positive or negative), so that $\mathrm{IC}_j(\mathbf{x})$ can only assume integer values. The linearity assumption of all the constraints and the requirement that the inequality ones are integer valued are necessary in order to introduce the standard method to obtain QUBO reformulations, as we show in section 2.1. Later, in section 3, we introduce novel methods that do not require the starting optimization problem to satisfy none of these assumptions: non-linear and non-integer constraints can be enforced without additional effort with respect to the linear integer-valued constraint cases.

2.1. Standard QUBO formulation

The goal of a QUBO reformulation of the optimization problem (2) is to provide a QUBO problem

$$\mathcal{E}^* = \arg\max_{\mathbf{x},\mathbf{s}} W(\mathbf{x}) + \lambda \left(\sum_{i=1}^{m_{\mathrm{EC}}} P_i^{\mathrm{EC}}(\mathbf{x}) + \sum_{j=1}^{m_{\mathrm{IC}}} P_j^{\mathrm{IC}}(\mathbf{x}, \mathbf{s}) \right) , \tag{3}$$

that provides the same optimal configurations provided by equation (2), where $\mathbf{s} = (s_1, s_2, \ldots, s_{\overline{N}})$ are $\overline{N}$ additional slack variables that are employed to enforce inequality constraints in a quadratic form. For the sake of clarity, we dropped the notation that makes explicit the dependence of W, EC_i and IC_j from w and $\mathbf{f}$. As we see later, linear equality constraints do not need any slack variables to be enforced in this form. In order for this reformulation to have the meaning that we required, we have to find quadratic forms P that provide penalties, namely are smaller or equal than -1, whenever the corresponding equality or inequality constraint are violated by $\mathbf{x}$, otherwise, if satisfied, they have to be zero valued for at least one value of $\mathbf{s}$. For instance, $P_1^{\mathrm{EC}}(\mathbf{x}) \leqslant -1$ if and only if $\mathbf{x}$ violates the first equality constraint and $P_1^{\mathrm{EC}}(\mathbf{x}) = 0$ if and only if $\mathbf{x}$ satisfies the same constraint. Here, $\lambda > 0$ is a multiplier that has to be chosen large enough in order to make sure that all the combinations of $\mathbf{x}$ violating one or more constraints cannot be the optimal configurations selected by equation (3).

The standard approach to enforce constraints in a quadratic form [28] reads as follows:

$$\mathcal{E}^* = \arg\max_{\mathbf{x},\mathbf{s}} W(\mathbf{x}) + \lambda \left(-\sum_{i=1}^{m_{\mathrm{EC}}} EC_i^2(\mathbf{x}) - \sum_{j=1}^{m_{\mathrm{IC}}} \left(\mathrm{IC}_j(\mathbf{x}) + S_j(\mathbf{s}) \right)^2 \right) , \tag{4}$$

where $S_j(\mathbf{s})$ are linear functions of the binary slack variables $\mathbf{s} = \{s_1, s_2, \ldots, s_{\overline{N}}\}$. The quadratic term that enforces the equality constraint $\mathrm{EC}_i(\mathbf{x})$ in the QUBO form is simply given by $P_i^{\mathrm{EC}}(\mathbf{x}) = -\mathrm{EC}_i^2(\mathbf{x})$. The reason of this choice is straightforward: any $\mathbf{x}$ that violates $\mathrm{EC}_i(\mathbf{x})$ provides a negative contribution. Therefore, as this optimization is formulated as a maximization, those $\mathbf{x}$ leading to a violation of an equality constraint are discouraged. Hence, if $\lambda > 0$ is set large enough, we are sure that the solution of equation (4) satisfies all the equality constraints.

For what concerns the inequality constraints, we build some $S_j(\mathbf{s})$ that help to translate these constraints into equality ones [28]. Before explaining how this goal is achieved, we notice that, if the inequality constraints are satisfied for at least one combination of $\mathbf{x}$, namely if there exists a solution of equation (2), then $\min_{\mathbf{x}} \mathrm{IC}_j(\mathbf{x}) \leqslant 0$. Hence, if $S_j(\mathbf{s})$ assumes all the values in the interval $[0, |\min_{\mathbf{x}} \mathrm{IC}_j(\mathbf{x})|]$, checking whether $\mathbf{x}$ satisfies the inequality constraint $\mathrm{IC}_j(\mathbf{x}) \leqslant 0$ is equivalent to check whether $\mathbf{x}$ satisfies the equality constraint $\mathrm{IC}_j(\mathbf{x}) + S_j(\mathbf{s}) = 0$ for at least one value of $\mathbf{s}$. Hence, the enforcement of this new equality constraint in the QUBO form can be made similarly to $\mathrm{EC}_i(\mathbf{x})$ by considering the quadratic term $P_j^{\mathrm{IC}}(\mathbf{x}, \mathbf{s}) = -(\mathrm{IC}_j(\mathbf{x}) + S_j(\mathbf{s}))^2$. Finally, in order for $S_j(\mathbf{s})$ to have the properties described above, we use the definition:

$$S_j(\mathbf{s}) = s_{\overline{N}_j} \overline{S}_j + \sum_{i=1}^{\overline{N}_j - 1} s_i 2^{i-1}, \tag{5}$$

where $\overline{N}_j > 0$ and $\overline{S}_j > 0$ are such that $S_j(\mathbf{s})$ assumes all the integer values in the interval $[0, |\min_{\mathbf{x}} \mathrm{IC}_j(\mathbf{x})|]$. Hence, $\overline{N}_j > 0$ is the number of slack variables needed by this method to enforce $\mathrm{IC}_j(\mathbf{x})$ in a QUBO form, namely equation (4). Notice that in equation (5) we implied a relabelling of the slack variable indices such that those associated to $\mathrm{IC}_j(\mathbf{x})$ are the first $\overline{N}_j > 0$, where $\overline{N} = \sum_j \overline{N}_j$. It is possible to apply similar techniques also in the case of non-integer-valued inequality constraints, but in general their enforcement requires more slack variables.

In summary, each term of the first and second sum of equation (4) is respectively associated to an equality or inequality constraint which provides a negative contribution, or penalty, if and only if $\mathbf{x}$ violates the corresponding condition. More precisely, if $\mathbf{x}$ violates a constraint, the contribution of the corresponding constraint term is strictly negative for all $\mathbf{s}$. On the other side, the maximizations over the same terms are zero-valued if and only if $\mathbf{x}$ satisfies the corresponding conditions. Hence, a large-enough value of $\lambda > 0$ guarantees that the optimal solution does not violate any constraint and therefore equations (2) and (4) have the same solution. Whereas a straightforward choice of this parameter is $\lambda > \max_{\mathbf{x}} W(\mathbf{x})$, we provide other techniques later in this work for this tuning.

We underline that, in order to achieve this QUBO formulation, we have to make a disjoint use of slack variables, namely each slack variable appearing in $S_j(\mathbf{s})$, does not appear in any other $S_k(\mathbf{s})$ for $j \neq k$. Moreover, the total number of slack variables $\overline{N}_j$ used to enforce $\mathrm{IC}_j(\mathbf{x})$ solely depends on $\min_{\mathbf{x}} \mathrm{IC}_j(\mathbf{x})$. As a result, the number of slack variables deployed is:

$$\overline{N} = \sum_{j=1}^{m_{\mathrm{IC}}} \overline{N}_j = \sum_{j=1}^{m_{\mathrm{IC}}} \left\lceil \log_2 \left(1 + \left| \min_{\mathbf{x}} \mathrm{IC}_j(\mathbf{x}) \right| \right) \right\rceil. \tag{6}$$

3. IQP and MS methods

The main goal of this work is to introduce a novel set of tools to obtain QUBO formulations of constrained binary optimization problems with the least possible amount of slack variables. This can help to reduce the number of qubits required to solve these problems on NISQ devices, potentially unlocking the possibility to tackle larger and more complex optimizations. Our strategy consists of two partially independent methods: the IQP method and the MS method. While here we provide an intuitive explanation of these techniques, a more detailed description can be found in the next sections.

The IQP method provides a new way to enforce constraints in a QUBO form. This method is particularly well-suited for constraints defined on few binary variables. Moreover, it can be easily generalized to enforce non-linear equality and inequality constraints into a QUBO form defined with integer or non-integer parameters. We summarize the main idea behind this method in figure 1, while its technical explanation is presented in section 3.1.

The second main contribution of this work is a novel interplay between the penalty quadratic forms of a QUBO problem that enforce constraints, called the MS method. While the standard approach enforces constraints independently from each other, our method exploits a synergy between different constraints sharing the same binary variables. In case of two or more such constraints, we can divide them into *master* and *satellite*. Master constraints are enforced for each possible binary variable combination, while satellite constraints are enforced only for those combinations that *already satisfy* the master constraints. As a result, the penalty quadratic forms associated with master constraints provide penalties/no-penalties whenever the original constraints are violated/satisfied, namely they perfectly enforce their corresponding constraints. Instead, the penalty quadratic forms associated with satellite

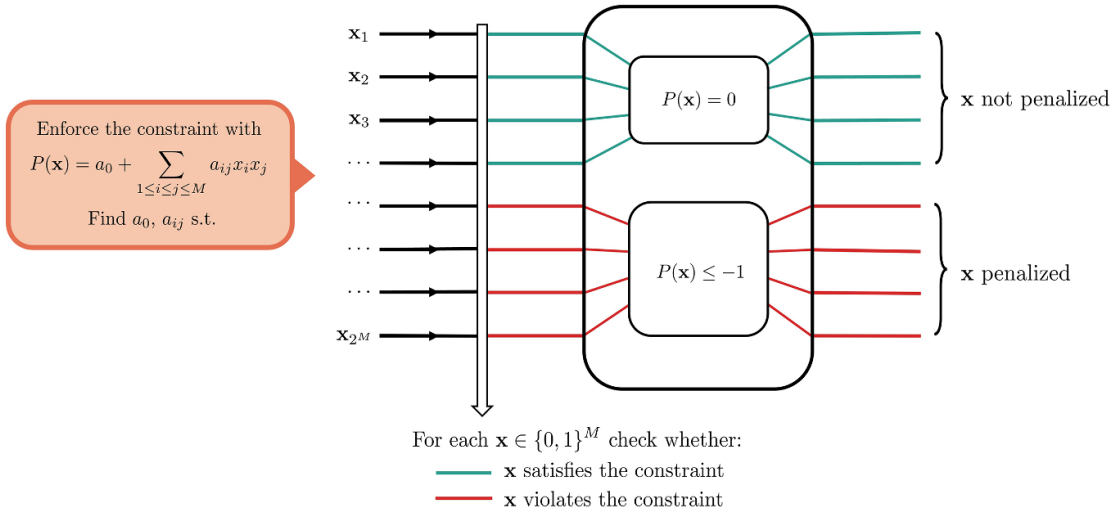


Figure 1. Iterative quadratic polynomial method. Take a constraint defined over M binary variables. Our goal is to enforce it in a QUBO form through the most generic quadratic polynomial $P(\mathbf{x})$ in the variables $\mathbf{x} = (x_1, x_2, \ldots, x_M)$. We look for the values of its free parameters a_0 and a_{ij} such that $P(\mathbf{x}) = 0$ for those combinations of $(x_1, x_2, \ldots, x_M)$ satisfying the constraint (green lines) and $P(\mathbf{x}) \leqslant -1$ for those combinations violating the constraint (red lines). In case such a polynomial does not exists, we iteratively increase the number of slack variables in order to have more and more free parameters available to define the polynomial. We give a detailed explanation of the IQP method (with and without slack variables) in section 3.1.

constraints are enforced only for those binary variables combinations that already satisfy the master constraints. A detailed description of the MS method is presented in section 3.2.

A consequence of this approach is that the quadratic forms associated to satellite constraints may provide *accidental incentives*, namely a positive contribution, for those binary variables combinations violating the master constraints. Hence, in order to avoid the promotion of those combinations, we amplify the influence of master constraints penalty terms until we have a net penalty even for those combinations receiving accidental incentives. This is done by a careful calibration of the QUBO penalty multipliers, namely some quantities analogous to the parameter λ from equation (4). This tuning is explained in detail in section 3.3. We show that, through the adoption of this strategy, we are able to reduce the employment of slack variables. We summarize the idea behind the MS method in figure 2.

It is evident that, given a particular optimization problem, the task of deciding which constraints have to be enforced as master and which as satellite is not a trivial one. Remember that, this method requires the presence of at least one master constraint and the more master constraints we have, the more efficient the implementation of satellite constraints is going to be. We can either decide to consider as master those constraints that require the least amount of slack variables, with or without the IQP method, or maybe those that are the most regular among the graph, e.g. do not depend on w and $\mathbf{f}$ but solely on the arcs and nodes in the multigraph. This situation is examined through an exemplary problem in section 5. We point out that, the separation of master and satellite constraints described here is called *bipartite*, namely when constraint can either be master or satellite. We introduce a *concatenated* version of the MS method in section 3.4, which results to be even more efficient in terms of slack variables employed.

The implementation of the MS method is particularly straightforward when satellite constraints are enforced with the IQP method. Indeed, this method gives us the possibility to decide whether to associate a penalty or not to precise binary variables combinations. Moreover, when satellite constraints are enforced with the IQP method, even less slack variables are necessary, if compared to the enforcement of the same constraints as master. Indeed, the slack variables needed by the IQP method increase with the number of binary variables combinations for which we require a penalty or not. Hence, we anticipate that the benefits of the conjunction between the IQP and the MS (IQPMS) methods are maximized when dealing with constraints defined on few common binary variables. Nonetheless, in appendix A we show how to apply the MS method when constraints are enforced with the standard method explained in section 2.1.

In summary, while the IQP method is more efficient for constraints defined over few variables, the MS approach can be applied for sets of constraints defined on the same subsets of binary variables.

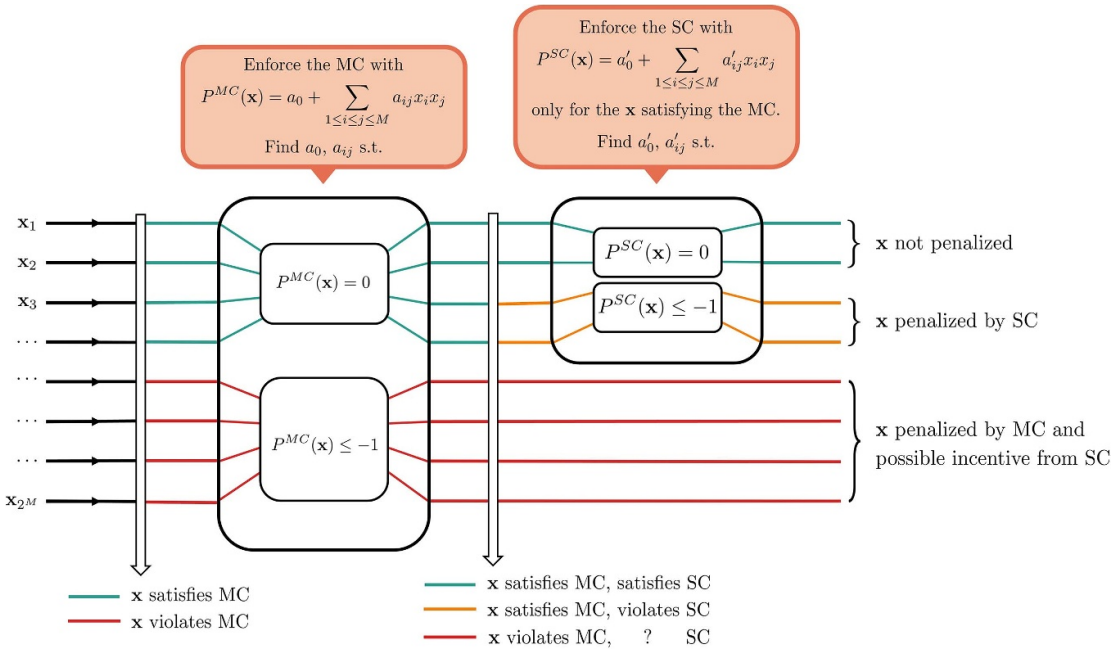


Figure 2. Master-satellite method. Take two constraints defined over the same M binary variables, where one is considered as master constraint (MC) and one as satellite constraint (SC). Our goal is to enforce them in a QUBO form through the most generic quadratic polynomials $P^{\mathrm{MC}}(\mathbf{x})$ and $P^{\mathrm{SC}}(\mathbf{x})$ in the variables $(x_1, x_2, \ldots, x_M)$. While the MC is enforced with the standard IQP method (see figure 1), the SC is enforced only for those combinations already satisfying the MC (top left green lines). Compared to $P^{\mathrm{MC}}(\mathbf{x})$, $P^{\mathrm{SC}}(\mathbf{x})$ is defined through a reduced number of conditions: no requirements are applied for those combinations violating the MC. As a consequence, we may obtain accidental incentives $P^{\mathrm{SC}}(\mathbf{x}) > 0$ for these combinations (red lines). We solve this issue by amplifying the role of $P^{\mathrm{MC}}(\mathbf{x})$ through a relative multiplier $\lambda^{\mathrm{MC}} > 1$ so that $\lambda^{\mathrm{MC}}P^{\mathrm{MC}}(\mathbf{x}) + P^{\mathrm{SC}}(\mathbf{x}) \leqslant -1$ if either the MC or the SC is violated (orange and red lines) and is equal to 0 if they are both satisfied (top right green lines). In case one or both such polynomials do not exists, we iteratively increase the number of slack variables in order to have more and more free parameters available to define the polynomials. We give a detailed explanation of the MS method in section 3.2.

Depending on the features of the combinatorial optimization problem studied, either one or both methods can be applied. For the sake of clarity, we introduce the class of locally-constrained optimizations problems in section 4, for which a simultaneous adoption of the IQPMS methods can be efficiently implemented, namely all constraints can be enforced with these new methods. Moreover, the IQPMS methods perform the best when the graphs defining these problems are lowly connected.

Finally, we point out that, while both the IQP and the MS methods offer several advantages, it is important to note their potential limitations. The IQP method can become computationally complex for constraints defined over several variables, and the effectiveness of the MS method may depend on the particular selection of master and satellite constraints. Nonetheless, compared to existing approaches in the literature, such as the alternating direction method of multipliers [32], the augmented Lagrangian method [33, 34] or approximate QUBO enforcing, which have recently been adopted in [35–40], our methods do not require solving preliminary QUBOs or approximating the problem constraints.

3.1. IQP method

In this section we introduce a novel technique to enforce constraints in the QUBO form. We start by focusing on the construction of a quadratic polynomial $P(\mathbf{x}, \mathbf{s})$ that enforces an inequality constraint $\mathrm{IC}(\mathbf{x}) \leqslant 0$ of the target optimization problem, see equation (2). As we explain later, this technique can also be used for more general constraints, but for the sake of simplicity we begin with the linear inequality case. We consider the case where the constraint $\mathrm{IC}(\mathbf{x}) = \mathrm{IC}(\mathbf{y})$ is defined over $M \leqslant N$ binary variables, namely over a subset $\mathbf{y} \subseteq \mathbf{x}$ of all the N binary variables $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ defining the starting optimization problem. Hence, differently from the previous section (see figure 1), we relabel the binary variables associated to the target inequality constraint by using $\mathbf{y} = (x_1, x_2, \ldots, x_M)$.

Consider a generic quadratic polynomial (QP) in the following $M + \overline{M}$ variables $\{\mathbf{y}, \mathbf{s}\}$, where $\mathbf{y} = (x_1, x_2, \ldots, x_M)$ are the binary variables associated to the studied constraint $\mathrm{IC}(\mathbf{y})$ and $\mathbf{s} = \{s_1, s_2, \ldots, s_{\overline{M}}\}$ are $\overline{M}$ slack variables:

$$P(\mathbf{y}, \mathbf{s}) = a_0 + \sum_{1\leqslant i\leqslant j\leqslant M} a_{ij}x_i x_j + \sum_{1\leqslant k\leqslant l\leqslant \overline{M}} \overline{a}_{kl}s_k s_l + \sum_{\substack{1\leqslant i\leqslant M \\ 1\leqslant k\leqslant \overline{M}}} b_{ik}x_i s_k\,. \tag{7}$$

The number of free parameters in $P(\mathbf{y},\mathbf{s})$ are:

$$\#\left\{a_0, a_{ij}, \overline{a}_{kl}, b_{ik}\right\} = 1 + \frac{\left(M+\overline{M}\right)^2 + M + \overline{M}}{2}. \tag{8}$$

The slack variables used to define $P(\mathbf{y},\mathbf{s})$ are disjoint from those considered to define QPs associated to other constraints [46].

QP with no-slack variables: We start by considering the scenario without slack variables, namely $\overline{M}=0$. In order to obtain a quadratic form $P(\mathbf{y})$ that enforces the constraint $\mathrm{IC}(\mathbf{y}) \leqslant 0$, we have to solve the following linear system in the parameters a_{ij}:

$$\begin{cases} P(\mathbf{y}) = 0 & \text{if } \mathbf{y} \text{ satisfies IC}(\mathbf{y}) \leqslant 0 \\ P(\mathbf{y}) \leqslant -1 & \text{if } \mathbf{y} \text{ violates IC}(\mathbf{y}) \leqslant 0 \end{cases}. \tag{9}$$

If such a solution exists, then we managed to translate the constraint $\mathrm{IC}(\mathbf{y}) \leqslant 0$ into a QUBO form without using slack variables. We say that each value of $\mathbf{y}$ identifies a different *sub-constraint*, which can be either assigned to a *no-penalty* $P(\mathbf{y}) = 0$ or to a *penalty* $P(\mathbf{y}) \leqslant -1$. Hence, in order for $P(\mathbf{y})$ to represent $\mathrm{IC}(\mathbf{y}) \leqslant 0$, it has to satisfy 2^M sub-constraints, one for each value of $\mathbf{y} \in \{0,1\}^M$. It may sound that we can use no slack variables only when $\#a_{ij} = 1 + (M^2+M)/2$ is not smaller than the number of sub-constraint 2^M. Nonetheless, as we see later, it is not uncommon to find quadratic polynomials that enforce a number of sub-constraints larger than the number of its free parameters. This is because inequality sub-constraints, namely $P(\mathbf{y}) \leqslant -1$, are less restrictive than equality ones. Finally, notice that, if the constraint is satisfied for all $\mathbf{y} \in \{0,1\}^M$, then we do not even need to enforce this constraint via a penalty polynomial: it is not violated for any combination of $\mathbf{y}$ and therefore no penalty has to be applied. Indeed, in this case we would simply obtain the null polynomial, namely $a_0 = a_{ij} = 0$ for all i and j.

QP with slack variables: The implementation of slack variables into this scenario cannot be made by simply considering the linear system (9), where $P(\mathbf{y})$ is replaced by $P(\mathbf{y},\mathbf{s})$. Indeed, even if $P(\mathbf{y},\mathbf{s})$ has several more degrees of freedom coming from the presence of slack variables, these are not exploited in the system as there is no reference of them in the linear system that we should solve. Consider for instance a value of $\mathbf{y}$ associated to a no-penalty sub-constraint. In this case, we would require $P(\mathbf{y},\mathbf{s}) = 0$ for all the values of $\mathbf{s}$. Hence, even if we have more degrees of freedom compared to the no-slack variable method, in this way we would solely have more conditions in the linear system. Therefore, it is easy to convince ourselves that this approach provides no improvement with respect to the scenario of a QP with no slack variables.

In order to exploit the potential of the slack variables inside a generic QP in this context, we start by noting the following. If we replace $P(\mathbf{y})$ with $P(\mathbf{y},\mathbf{s})$ in equation (9), the penalty (inequality) sub-constraints are less restrictive than the no-penalty (equality) ones. Indeed, the former type allows $P(\mathbf{y},\mathbf{s})$ to assume any value smaller or equal to -1 and for different values of $\mathbf{s}$ and the same $\mathbf{y}$ the polynomial $P(\mathbf{y},\mathbf{s})$ can assume different penalties. Instead, for no-penalty sub-constraints we require $P(\mathbf{y},\mathbf{s})$ to be exactly 0, for all $\mathbf{y}$ and $\mathbf{s}$. Nonetheless, if $\mathbf{y}$ is associated to a no-penalty, we do not need $P(\mathbf{y},\mathbf{s}) = 0$ to be true *for all the values* of $\mathbf{s}$, it is enough to require it for *at least one* $\mathbf{s}$, while at the same time no incentives are provided for the other values of $\mathbf{s}$. Indeed, $P(\mathbf{y},\mathbf{s})$ is one component of the QUBO formulation of an optimization problem, which undergoes a maximization over $\mathbf{x}$ (which contains all the variables in $\mathbf{y}$) and $\mathbf{s}$. Hence, our minimal requirements on the parameters a_{ij}, $\overline{a}_{kl}$ and b_{ik} defining $P(\mathbf{y},\mathbf{s})$ are:

$$\begin{cases} P(\mathbf{y},\mathbf{s}) \leqslant -1 & \text{if } \mathbf{y} \text{ violates IC}(\mathbf{y}) \leqslant 0 \quad \text{for all } \mathbf{s} \\ P(\mathbf{y},\mathbf{s}) \leqslant 0 & \text{if } \mathbf{y} \text{ satisfies IC}(\mathbf{y}) \leqslant 0 \quad \text{for all } \mathbf{s} \\ P(\mathbf{y},\mathbf{s}) = 0 & \text{if } \mathbf{y} \text{ satisfies IC}(\mathbf{y}) \leqslant 0 \quad \text{for at least one } \mathbf{s} \end{cases} \tag{10}$$

or, equivalently,

$$\begin{cases} P(\mathbf{y},\mathbf{s}) \leqslant -1 & \text{if } \mathbf{y} \text{ violates IC}(\mathbf{y}) \leqslant 0 \text{ for all } \mathbf{s} \\ \max_{\mathbf{s}} P(\mathbf{y},\mathbf{s}) = 0 & \text{if } \mathbf{y} \text{ satisfies IC}(\mathbf{y}) \leqslant 0 \end{cases}. \tag{11}$$

The enforcement of the sub-constraints proposed in equations (10) and (11) can be casted as a *feasibility problem*. In case of a single slack variable $\mathbf{s} = s_1$, namely for $\overline{M}=1$, it assumes the form:

$$\begin{array}{lll} \arg\ \min_{a_0, a_{ij}, \overline{a}_{kl}, b_{ik}} & 1 & \\ \text{s.t.} & P(\mathbf{y},0) \leqslant -1 \wedge P(\mathbf{y},1) \leqslant -1 & \text{if } \mathbf{y} \text{ violates IC}(\mathbf{y}) \leqslant 0 \\ & (P(\mathbf{y},0) = 0 \wedge P(\mathbf{y},1) \leqslant 0) \vee (P(\mathbf{y},0) \leqslant 0 \wedge P(\mathbf{y},1) = 0) & \text{if } \mathbf{y} \text{ satisfies IC}(\mathbf{y}) \leqslant 0 \end{array}, \tag{12}$$

where the symbols $\wedge$ and $\vee$ represent, respectively, the AND and OR logical operations. Hence, the solution of this feasibility problem provides the parameters $a_{ij}, \bar{a}_{kl}$ and b_{ik} that defines the quadratic polynomial $P(\mathbf{y}, s_1)$ that enforces our constraint.

IQP method: Consider the following iterative procedure to enforce constraints in a QUBO form, where in each step we add a slack variable to the QP used, until this enforcement is possible. Hence, given a target constraint, we execute the following step-by-step algorithm:

- STEP 0: Apply the method with no-slack variables given in equation (9). If it provides a solution, halt.
- STEP 1: Apply the method with $\overline{M} = 1$ slack variable given in equations (10) and (11). If it provides a solution, halt.
- STEP 2: Apply the method with $\overline{M} = 2$ slack variables given in equations (10) and (11). If it provides a solution, halt.
- …
- STEP $\overline{M}$: Apply the method with $\overline{M}$ slack variables given in equations (10) and (11), with the following feasibility form:

$$\begin{array}{lll} \arg & \min_{a_0, a_{ij}, \bar{a}_{kl}, b_{ik}} & 1 \\ & \text{s.t.} & \bigwedge_{\mathbf{s}} P(\mathbf{y}, \mathbf{s}) \leqslant -1 \quad \text{if } \mathbf{y} \text{ violates IC}(\mathbf{y}) \leqslant 0 \\ & & \bigvee_{\mathbf{s}} \left(P(\mathbf{y}, \mathbf{s}) = 0 \wedge P(\mathbf{y}, \mathbf{s}' \neq \mathbf{s}) \leqslant 0 \right) \quad \text{if } \mathbf{y} \text{ satisfies IC}(\mathbf{y}) \leqslant 0 \end{array} , \tag{13}$$

where the AND and OR logical operations are performed over $\mathbf{s} \in \{0,1\}^{\overline{M}}$.

In the following, we use the expression 'IQP method' to refer to this algorithm. Notice that, if this procedure halts at a certain step, then the same amount of slack variables are enough for the IQP method to encode the target constraint as a QUBO penalty term. Remember that we start from STEP 0, which corresponds to the no-slack variables scenario. Hence, this method is designed to enforce our constraint via a QP (7) employing the least number of slack variables. In the following, the QP obtained through this method could be simply called IQP.

We point out that, as we increase the number of slack variables throughout these steps, $P(\mathbf{y}, \mathbf{s})$ gets more and more free-parameters at disposal (see equation (8)), while at the same time we apply the same amount of 'strong' equality conditions and more 'soft' inequality conditions (see equation (10)) in the corresponding linear system. An indicative estimate of the slack variables necessary for this kind of method is given by requiring that the free parameters in $P(\mathbf{y}, \mathbf{s})$ (see equation (8)) is larger than the number of sub-constraints, which in the general case are 2^M, namely the conditions in the linear system (9) considered with no slack variables. Nonetheless, as we show it later with an example, a lower number of slack variables is often sufficient. Hence, it should already be clear that this method is particularly useful for small M. Indeed, while for the standard method the number of slack variables do not change with M, in general the number of slack variables needed by the IQP method increases with M.

Finally, we underline two more qualities of this technique. First, this method can be extended to any linear/non-linear equality/inequality constraint defined with integer/non-integer parameters. Indeed, consider the linear systems (10) and (13) defining the parameters of the quadratic polynomial enforcing the constraint. The proposed procedure makes no reference to the form of IC($\mathbf{y}$) and, more importantly, its linearity is never exploited. The only operation that we have to perform is to verify whether IC($\mathbf{y}$) is violated or not for each $\mathbf{y}$. It follows that, in case we need to enforce any other type of constraint $C(\mathbf{y}) = \text{True}$, the same procedure described in this section can be adopted, where we simply replace '$\mathbf{y}$ satisfies/violates IC($\mathbf{y}$) $\leqslant 0$' with '$\mathbf{y}$ satisfies/violates $C(\mathbf{y})$'. A second merit of the IQP method is that, differently from the standard approach, it does not require all the variables to be coupled to each other: some of the parameters $a_{ij}, \bar{a}_{kl}, b_{ik}$ can be null. We encounter this scenario for several constraints defining the optimization problem studied later in this work, namely the MPBS problem (see section 6). This scenario can improve the performance of NISQ devices used as QUBO solvers as sparser formulations require fewer and less-connected qubits.

3.2. MS methods

Consider the scenario where two (or more) constraints are defined over the same subset of variables $\mathbf{y} \subseteq \mathbf{x}$, where the first has been enforced in a quadratic form either with the IQP method or any other method, e.g. as in section 2.1. We call this constraint the master constraint, while the second, which has yet to be enforced, as satellite. We follow by presenting a variation of the IQP method suited to enforce satellite constraints. This technique, as we show below, employs even less slack variables, if compared with the method presented in section 3.1, which instead is better suited for master constraints.

The main difference between the standard IQP method and its variation for satellite constraints is that we do not impose sub-constraints for all the possible combinations of the binary variables $\mathbf{y} = (x_1, x_2, \ldots, x_M)$, but only for the strictly smaller subset of combinations satisfying the master constraints. The final discussion of the previous section should already convince the reader that having less than 2^M sub-constraints allows to implement satellite constraints with even fewer slack variables.

We start by considering two constraints sharing the same variables $\mathbf{y}$, where we call 'MC$(\mathbf{y}) =$ True' the constraint considered as master, and 'SC$(\mathbf{y}) =$ True' the satellite one, where both can either be linear/non-linear equality/inequality constraints defined with integer/non-integer parameters. Requiring $P^{\mathrm{SC}}(\mathbf{y}, \mathbf{s})$, obtained form equation (7), to enforce the satellite constraint with $\overline{M}$ slack variables corresponds to verify whether there exists a solution of the following linear system in the parameters a_{ij}, $\bar{a}_{kl}$ and b_{ik}:

$$\begin{cases} P^{\mathrm{SC}}(\mathbf{y}, \mathbf{s}) \leqslant -1 & \text{if } \mathbf{y} \text{ satisfies MC}(\mathbf{y}) \text{ and violates SC}(\mathbf{y}) \quad \text{for all } \mathbf{s} \\ P^{\mathrm{SC}}(\mathbf{y}, \mathbf{s}) \leqslant 0 & \text{if } \mathbf{y} \text{ satisfies MC}(\mathbf{y}) \text{ and satisfies SC}(\mathbf{y}) \quad \text{for all } \mathbf{s} \\ P^{\mathrm{SC}}(\mathbf{y}, \mathbf{s}) = 0 & \text{if } \mathbf{y} \text{ satisfies MC}(\mathbf{y}) \text{ and satisfies SC}(\mathbf{y}) \quad \text{for at least one } \mathbf{s} \end{cases} \tag{14}$$

or, equivalently,

$$\begin{cases} P^{\mathrm{SC}}(\mathbf{y}, \mathbf{s}) \leqslant -1 & \text{if } \mathbf{y} \text{ satisfies MC}(\mathbf{y}) \text{ and violates SC}(\mathbf{y}) \quad \text{for all } \mathbf{s} \\ \max_{\mathbf{s}} P^{\mathrm{SC}}(\mathbf{y}, \mathbf{s}) = 0 & \text{if } \mathbf{y} \text{ satisfies MC}(\mathbf{y}) \text{ and satisfies S C}(\mathbf{y}) \end{cases} . \tag{15}$$

The step-by-step IQP method described in the previous section can be generalized to the satellite scenario, where we simply restrict the corresponding conditions for $P^{\mathrm{SC}}(\mathbf{y}, \mathbf{s})$ to the $2^M - n(\mathrm{MC}) < 2^M$ sub-constraints that satisfy the master constraints, as shown in equations (14) and (15), where $n(\mathrm{MC}) \geqslant 1$ is the number of combinations of $\mathbf{y}$ that violate the master constraint. Hence, all the corresponding steps can be formulated similarly. For instance, the feasibility problem formulation (13) reads:

$$\begin{array}{lll} \arg \min_{a_{ij}, \bar{a}_{kl}, b_{ik}} & 1 & \\ \text{s.t.} & \bigwedge_{\mathbf{s}} P^{\mathrm{SC}}(\mathbf{y}, \mathbf{s}) \leqslant -1 & \text{if } \mathbf{y} \text{ satisfies MC}(\mathbf{y}) \text{ and violates SC}(\mathbf{y}) \\ & \bigvee_{\mathbf{s}} P^{\mathrm{SC}}(\mathbf{y}, \mathbf{s}) = 0 \wedge P^{\mathrm{S\,C}}(\mathbf{y}, \mathbf{s}' \neq \mathbf{s}) \leqslant 0 & \text{if } \mathbf{y} \text{ satisfies MC}(\mathbf{y}) \text{ and satisfies SC}(\mathbf{y}) \end{array} , \tag{16}$$

where the AND and OR logical operations are performed over $\mathbf{s} \in \{0,1\}^{\overline{M}}$. Remember that this procedure can be straightforwardly applied to constraints that are non-linear or not defined through integer parameters, as described in section 3.1.

Notice that, in case all the combinations of $\mathbf{y}$ satisfying the master constraint satisfy the satellite constraint too, then there is no need to enforce the satellite constraint with a quadratic polynomial at all. Indeed, in this case a solution of equation (16) would be given (without employing any slack variable) by $a_0 = a_{ij} = \bar{a}_{kl} = b_{ik} = 0$ for all i, j, k and l.

In order to make the notation clearer, we considered the case of one master and one satellite constraint. In case of multiple master constraints, we have to require sub-constraints only for those $\mathbf{y}$ satisfying *all* the master constraints. Hence, in such a scenario we would impose even less sub-constraints, namely $2^M - n(\mathrm{MC}) < 2^M$, where now $n(\mathrm{MC}) \geqslant 1$ is the number of combinations that violate at least one master constraint. Naturally, if we have more than one satellite constraint, the solution of this linear system has to be performed independently for each one of these constraints.

The natural consequence of not imposing any sub-constraint for those $\mathbf{y}$ violating the master constraints is that we may obtain an accidental incentive $P^{\mathrm{SC}}(\mathbf{y}, \mathbf{s}) > 0$ for some $(\mathbf{y}, \mathbf{s})$, where $\mathbf{y}$ violates the master constraint. In this situation such a combination would be promoted, and therefore we need a strategy to avoid that the optimal solution of the corresponding QUBO problem violates one or more constraints. This goal is achieved by amplifying the weight of master constraints as shown in section 3.3.

We say that this MS method adopts a *bipartite* approach as, even in case of more than two constraints involved, each constraint is either considered as a master or a satellite constraint. Instead, in case of three or more constraints, we can either adopt the strategy introduced in this section or its *concatenated* version, presented in section 3.4.

3.3. Tuning of the relative multipliers

Consider the QUBO form (3) of a generic optimization problem (2), where a single constraint has been enforced with the IQP method presented in section 3.1. Since the corresponding quadratic form $P(\mathbf{x}, \mathbf{s}) = P(\mathbf{y}, \mathbf{s})$ provides penalties/no-penalties if and only if $\mathbf{y}$ violates/satisfies the corresponding constraint, no adjustments have to be applied to the form given in equation (3). The same applies if

multiple constraints are enforced with the standard IQP method, namely when the MS method is not exploited.

In case two or more constraints are enforced with the MS method, some adjustments to equation (3) have to be applied. In practice, we need to amplify the role of the master constraint quadratic form as follows:

$$P^{\mathrm{MC}}(\mathbf{y},\mathbf{s}) + P^{\mathrm{SC}}(\mathbf{y},\mathbf{s}) \longrightarrow \lambda^{\mathrm{MC}} P^{\mathrm{MC}}(\mathbf{y},\mathbf{s}) + P^{\mathrm{SC}}(\mathbf{y},\mathbf{s}'), \tag{17}$$

where the master and satellite constraints considered here are two constraints picked from the inequality and equality constraints of equation (2) that we decided to enforce with the MS approach, hence sharing the same subset of binary variables $\mathbf{y}$. Hence, $\lambda^{\mathrm{MC}} P^{\mathrm{MC}}(\mathbf{y},\mathbf{s}) + P^{\mathrm{SC}}(\mathbf{y},\mathbf{s}')$ replaces two of the quadratic polynomials on the r.h.s. of equation (3). The multiplier $\lambda^{\mathrm{MC}} \geqslant 1$ can be defined as follows:

$$\lambda^{\mathrm{MC}} = 1 + \gamma \max\left\{0, \max_{\mathbf{y},\mathbf{s}'} P^{\mathrm{SC}}(\mathbf{y},\mathbf{s}')\right\}, \tag{18}$$

where $\gamma > 1$ and, if the maximization on the r.h.s. is greater than zero, we say that at least one of the satellite constraints provide an *accidental incentive*. In case of no accidental incentives, $\lambda^{\mathrm{MC}} = 1$.

Thanks to this transformation, we finally achieve a QUBO form of these two constraints that faithfully represents the corresponding constraints. Indeed, the r.h.s. of equation (17) is smaller or equal than -1 if $\mathbf{y}$ violates either the master or the satellite constraint and is zero-valued for at least one value of $\mathbf{s}$ if both constraints are satisfied:

$$\begin{cases} \lambda^{\mathrm{MC}} P^{\mathrm{MC}}(\mathbf{y},\mathbf{s}) + P^{\mathrm{SC}}(\mathbf{y},\mathbf{s}') \leqslant -1 & \text{for all } \mathbf{s} & \text{if } \mathbf{y} \text{ violates } \mathrm{MC}(\mathbf{s}) \vee \mathrm{SC}(\mathbf{s}) \\ \lambda^{\mathrm{MC}} P^{\mathrm{MC}}(\mathbf{y},\mathbf{s}) + P^{\mathrm{SC}}(\mathbf{y},\mathbf{s}') \leqslant 0 & \text{for all } \mathbf{s} & \text{if } \mathbf{y} \text{ satisfies } \mathrm{MC}(\mathbf{s}) \wedge \mathrm{SC}(\mathbf{s}) \\ \lambda^{\mathrm{MC}} P^{\mathrm{MC}}(\mathbf{y},\mathbf{s}) + P^{\mathrm{SC}}(\mathbf{y},\mathbf{s}') = 0 & \text{for at least one } \mathbf{s} & \text{if } \mathbf{y} \text{ satisfies } \mathrm{MC}(\mathbf{s}) \wedge \mathrm{SC}(\mathbf{s}) \end{cases}. \tag{19}$$

Consider the case where we enforce multiple constraints as master and satellite (either with the IQP method or not) defined over the same shared set of variables $\mathbf{y} \subseteq \mathbf{x}$. Both master and satellite constraints can be chosen among the equality and inequality constraints. We apply the following transformation of the QUBO form:

$$\sum_{i=1}^{m_{\mathrm{MC}}} P_i^{\mathrm{MC}}(\mathbf{y},\mathbf{s}) + \sum_{j=1}^{m_{\mathrm{SC}}} P_j^{\mathrm{SC}}(\mathbf{y},\mathbf{s}) \longrightarrow \lambda^{\mathrm{MC}} \sum_{i=1}^{m_{\mathrm{MC}}} P_i^{\mathrm{MC}}(\mathbf{y},\mathbf{s}) + \sum_{j=1}^{m_{\mathrm{SC}}} P_j^{\mathrm{SC}}(\mathbf{y},\mathbf{s}), \tag{20}$$

where, even if not made explicit, the slack variables employed for the master and satellite quadratic polynomials are independent. Therefore, this transformation involves $m_{\mathrm{MC}} + m_{\mathrm{IC}}$ quadratic polynomials picked from equation (3). Similarly to equation (18), we use:

$$\lambda^{\mathrm{MC}} = 1 + \gamma \max\left\{0, \max_{\mathbf{y},\mathbf{s}'} \sum_{j=1}^{m_{\mathrm{SC}}} P_j^{\mathrm{SC}}(\mathbf{y},\mathbf{s}')\right\}. \tag{21}$$

For what concerns the multiplier λ that appears in equation (3), we can apply the ordinary strategies that can be found in the literature. Several exemplary problems for which this parameter is calculated are given in [5], where at times a different parameter λ_i is associated to each quadratic penalty term $P_i(\mathbf{x},\mathbf{s})$. In what follows, we define a class of optimization problems, called locally-constrained, for which we propose a different tuning of these parameters.

3.4. MS concatenation

We briefly describe a method to concatenate the MS approach in order to reduce even more the number of considered sub-constraints, and therefore of slack variables. This approach can be applied solely for those cases where three or more constraints share the same subset of variables $\mathbf{y} \subseteq \mathbf{x}$. For instance, consider the four constraints $C_1(\mathbf{y})$, $C_2(\mathbf{y})$, $C_3(\mathbf{y})$ and $C_4(\mathbf{y})$ (we omit the '= True' notation) defined over the same binary variables.

The approach of the previous sections requires to identify two groups of constraints, where all the constraints in the second group are satellite of the constraints in the first group. For instance, we could choose the master constraints to be $C_1(\mathbf{y})$ and $C_2(\mathbf{y})$, while their satellite constraints are respectively $C_3(\mathbf{y})$ and $C_4(\mathbf{y})$. Each constraint in the satellite group is actively enforced solely for those combinations of $\mathbf{y}$ that satisfy the constraints from the master group. On the other side, all the master constraints have to be enforced for all the 2^M combinations of $\mathbf{y} = (x_1, x_2, \ldots, x_M)$.

We concatenate the MS method among all the constraints defined over $\mathbf{y} \subseteq \mathbf{x}$. In particular, we start by choosing two constraints, e.g. $C_1(\mathbf{y})$ and $C_2(\mathbf{y})$, and enforce $C_1(\mathbf{y})$ *as the master constraint of* $C_2(\mathbf{y})$. Hence, $C_1(\mathbf{y})$ would be enforced for all the combinations of $\mathbf{y}$, while $C_2(\mathbf{y})$ only for those $\mathbf{y}$ satisfying $C_1(\mathbf{y})$ (as the MS approach described in the previous sections in the case of two constraints). Nonetheless, differently from before, we continue this concatenation of constraints by considering $C_1(\mathbf{y}) \wedge C_2(\mathbf{y})$ as the master constraint of $C_3(\mathbf{y})$. Hence, $C_3(\mathbf{y})$ would be enforced only for those $\mathbf{y}$ satisfying $C_1(\mathbf{y}) \wedge C_2(\mathbf{y})$. Finally, $C_4(\mathbf{y})$ would be enforced as a satellite constraint of all the other constraints, namely with $C_1(\mathbf{y}) \wedge C_2(\mathbf{y}) \wedge C_3(\mathbf{y})$ being its master constraint. Therefore, $C_4(\mathbf{y})$ has to be enforced for those $\mathbf{y}$ already satisfying all the other constraints. This hierarchy of constraint enforcement allows to consider fewer sub-constraints if compared with the non-concatenated case, and therefore, as show before, less slack variables.

This approach requires the introduction of multiple relative multipliers that have the role to counterbalance the accidental incentives coming from the enforcement of $C_2(\mathbf{y})$, $C_3(\mathbf{y})$ and $C_4(\mathbf{y})$. If we consider the previously described example of four constraints, we would need three different multipliers similar to λ^{MC} from equation (18) via the following replacement of the quadratic polynomials enforcing $C_{1,2,3,4}(\mathbf{y})$ in equation (3):

$$P_1(\mathbf{y},\mathbf{s}) + P_2(\mathbf{y},\mathbf{s}) + P_3(\mathbf{y},\mathbf{s}) + P_4(\mathbf{y},\mathbf{s}) \longrightarrow \lambda_1 P_1(\mathbf{y},\mathbf{s}) + \lambda_2 P_2(\mathbf{y},\mathbf{s}) + \lambda_3 P_3(\mathbf{y},\mathbf{s}) + P_4(\mathbf{y},\mathbf{s}) \,, \tag{22}$$

where $P_i(\mathbf{y},\mathbf{s})$ is the IQP obtained while considering the constraint $C_i(\mathbf{y})$ in this concatenated scheme, namely as a satellite of all the constraints $C_j(\mathbf{y})$ with $j<i$ and master of those with $j>i$. Remember that each $C_i(\mathbf{y})$ can either be an equality or an inequality constraint. The tuning of the parameters λ_i is given by the following generalization of equation (18):

$$\lambda_i = 1 + \gamma \max\left\{0, \max_{\mathbf{y},\mathbf{s}} \sum_{j>i} P_j(\mathbf{y},\mathbf{s})\right\} , \tag{23}$$

where $\gamma > 1$.

4. Locally-constrained optimization problems

A class of optimization problems that is particularly suited for the application of our techniques is given by the locally-constrained ones, namely those having constraints that are defined node-by-node and depend solely on the variables x_i associated to the arcs incident to each node (see definition 1). The optimization problems of this type assume the form:

$$\begin{aligned} \mathcal{E}^* = \arg \quad & \max_{\mathbf{x}} \quad W(\mathbf{x}) \\ & \text{s.t.} \quad \mathrm{EC}_{u,i}(\mathbf{x}) = 0 \quad \text{for } u \in \mathcal{V}(\mathbf{x}) \text{ and } \; i = 1,\ldots,m_{\mathrm{EC}} \\ & \qquad\; \mathrm{IC}_{u,j}(\mathbf{x}) \leqslant 0 \quad \text{for } u \in \mathcal{V}(\mathbf{x}) \text{ and } \; j = 1,\ldots,m_{\mathrm{IC}} \end{aligned} \,, \tag{24}$$

where the equality and inequality constraints $\mathrm{EC}_{u,i}(\mathbf{x})$ and $\mathrm{IC}_{u,j}(\mathbf{x})$ are defined solely over those binary variables $\mathbf{y}_u \subseteq \mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ associated to the arcs incident to the corresponding node u and $\mathcal{V}(\mathbf{x}) \subseteq \mathcal{V}$ is the subset of nodes identified by the arcs selected by $\mathbf{x}$. For instance, if $\mathbf{x} = \{1,0,0,\ldots,0\}$, where $x_1 = 1$ corresponds to an arc connecting $u=1$ and $u=2$, $\mathcal{V}(\mathbf{x})$ would consists of the nodes $u=1$ and $u=2$. Hence, for each node in $\mathcal{V}(\mathbf{x})$ we impose m_{EC} equality constraints and m_{IC} inequality constraints [47]. Notice that, while in equation (2) m_{EC} and m_{IC} referred to the total number of equality and inequality constraints in the whole graph, respectively, in equation (24) they refer to the number of equality/inequality constraints per node [48].

Locally-constrained optimization problems are particularly appropriate for the application of the IQPMS methods. These problems present a natural subdivision of the constraints defined over the same set of binary variables $\mathbf{y}$. Indeed, each node u has constraints defined solely over the variables $\mathbf{y}_u \subseteq \mathbf{x}$ associated to the arcs incident to u. Therefore, the application of the IQPMS methods is extremely direct.

Consider a locally-constrained problem with at least two constraints per node and divide them into $m_{\mathrm{MC}} \geqslant 1$ master constraints $MC_{u,i}(\mathbf{x})$ and $m_{\mathrm{SC}} \geqslant 1$ satellite constraints $SC_{u,j}(\mathbf{x})$, where $m_{\mathrm{MC}} + m_{\mathrm{SC}} = m_{\mathrm{EQ}} + m_{\mathrm{IQ}}$. All the linear equality constraints, since they do not need the employment of slack variables, are promoted to master constraints, while the remaining ones can either be satellite or master. We can adopt the IQPMS methods introduced in sections 3.1 and 3.2 to enforce the master and satellite constraints as quadratic penalty terms $P^{\mathrm{MC}}_{u,i}(\mathbf{y}_u,\mathbf{s})$ and $P^{\mathrm{SC}}_{u,j}(\mathbf{y}_u,\mathbf{s})$, which are defined over the variables

$\mathbf{y}_u \subseteq \mathbf{x}$ associated to the arcs incident to the node u. Therefore, the QUBO formulation of the optimization problems in this class, thanks to the IQPMS methods, assume the form:

$$\mathcal{E}^* = \arg\max_{\mathbf{x},\mathbf{s}} W(\mathbf{x}) + \sum_{u=1}^{|\mathcal{V}|} \lambda_u \left(\lambda_u^{\mathrm{MC}} \sum_{i=1}^{m_{\mathrm{MC}}} P_{u,i}^{\mathrm{MC}}(\mathbf{y}_u, \mathbf{s}) + \sum_{j=1}^{m_{\mathrm{SC}}} P_{u,j}^{\mathrm{SC}}(\mathbf{y}_u, \mathbf{s}) \right), \tag{25}$$

where the quantities $\lambda_{u,i}^{\mathrm{MC}} \geqslant 1$ and $\lambda_u \geqslant 1$ have two different roles. The relative multipliers $\lambda_{u,i}^{\mathrm{MC}}$ ensure that, for each combination of the binary variables associated to the node u, if $\mathbf{y}_u$ violates one of the constraints, the net contribution of the term inside the parenthesis is smaller than or equal to -1. Indeed, we need to amplify the effect of master constraints accordingly to the accidental incentives that $P_{u,j}^{\mathrm{SC}}(\mathbf{y}_u, \mathbf{s})$ provide. Instead, if $\mathbf{y}_u$ satisfies the constraints, the term inside the parenthesis has to be zero for at least one value of $\mathbf{s}$ and non-positive for the remaining combinations of $\mathbf{s}$. These quantities have the same role of the parameter λ^{MC} defined in equations (18) and (20). Instead, the parameters λ_u replace the role of λ in equation (3). Hence, they have to amplify enough the penalties in order to suppress those $\mathbf{y}_u$ that violate one or more constraints.

4.1. Tuning of the relative multipliers and the MS concatenation

We first discuss the tuning of the relative multipliers λ_u^{MC} and later we focus on λ_u. A straightforward application of the results presented in section 3.3 allows us to consider the following definition

$$\lambda_u^{\mathrm{MC}} = 1 + \gamma \max\left\{ 0, \max_{\mathbf{y}_u,\mathbf{s}} \sum_{j=1}^{m_{\mathrm{SC}}} P_{u,j}^{\mathrm{SC}}(\mathbf{y}_u, \mathbf{s}) \right\}, \tag{26}$$

where $\gamma > 1$. Notice that all the relative multipliers of the same node u have the same value.

In what follows, we describe three proposals for the tuning of λ_u, which we call *local, neighbor* and *global*. Our first approach is to choose a value of λ_u such that the minimum penalty applied when at least one constraint in u is violated is (in modulo) greater than the maximum contribution provided by the arcs incident in u to $W(\mathbf{x})$. In order to do this, we define $W_u(\mathbf{x})$ to be the restriction of $W(\mathbf{x})$ to the node u. More precisely, if we write $W(\mathbf{x}) = \sum_{1 \leqslant i \leqslant j \leqslant N} W_{ij} x_i x_j$, in $W_u(\mathbf{x})$ we include only those terms where either x_i, x_j or both belong to $\mathbf{y}_u$, namely corresponds to arcs in $\mathcal{E}(u)$. Hence, we define:

$$\lambda_u^{\mathrm{local}} = \gamma \max_{\mathbf{x}} W_u(\mathbf{x}), \tag{27}$$

where $\gamma > 1$.

The next two approaches that we define naturally provide larger values of λ_u. On the one side it helps preventing the 'non-local' violation of constraints that may occur in presence of a local tuning. In appendix C we describe this phenomenon in the context of the Max-Profit Balanced Settlement optimization problem: an NP-hard problem that we analyse in section 5. Nonetheless, the presence of penalty terms that are too large compared to those found in $W(\mathbf{x})$ can suppress the output quality of several QUBO solvers, especially quantum ones. Moreover, depending on the structure of $W(\mathbf{x})$, the evaluation of these two extra methods may be more demanding (this is not the case for diagonal $W(\mathbf{x})$). Therefore, in this work we privilege the use of the local tuning, and, in case of necessity, we pick larger values of γ.

The second possible tuning that we discuss extends the previous approach to those nodes neighboring u, namely those sharing an arc with u. Similarly to $W_u(\mathbf{x})$, we now define $W_u^{\mathrm{neigh}}(\mathbf{x})$ to be the restriction of $W(\mathbf{x})$ to the binary variables associated to arcs incident to u and its neighboring nodes. Hence, we define:

$$\lambda_u^{\mathrm{neigh}} = \gamma \max_{\mathbf{x}} W_u^{\mathrm{neigh}}(\mathbf{x}), \tag{28}$$

where $\gamma > 1$.

The last discussed tuning is labeled as global because it is not node-dependent and it makes sure that the penalty applied is always large enough whenever any constraint is violated. Indeed, it consists in setting:

$$\lambda_u^{\mathrm{global}} = \gamma \max_{\mathbf{x}} W(\mathbf{x}), \tag{29}$$

where $\gamma > 1$. Whereas this is a sure choice to make our QUBO formulation exactly encoding the starting optimization problem, namely they provide the same solution, it has some downturns. Depending on the structure of $W(\mathbf{x})$, the optimization $\max_{\mathbf{x}} W(\mathbf{x})$ may have a computational cost that is similar to the

whole optimization problem, and therefore evaluating $\lambda_u^{\text{global}}$ may become excessively demanding. More importantly, this choice can easily produce a QUBO problem where the penalty terms are (in modulo) extremely larger than the objective function terms. This scenario can deeply affect the solution quality of QUBO solvers and, moreover, we expect that in many common situations this brute-force approach is in fact unnecessary.

To conclude this section, we give an exemplary reformulation of the results regarding the concatenated IQPMS methods to the case of locally-constrained problems. Imagine to have four constraints per node, hence defined over the same binary variables $\mathbf{y}_u \subseteq \mathbf{x}$, and concatenate them as in section 3.4. We can reformulate the QUBO form (25) of this locally-constrained optimization problem as:

$$\mathcal{E}^* = \arg\max_{\mathbf{x},\mathbf{s}} W(\mathbf{x}) + \sum_{u=1}^{|\mathcal{V}|} \lambda_u \left(\lambda_{u,1} P_{u,1}(\mathbf{y}_u, \mathbf{s}) + \lambda_{u,2} P_{u,2}(\mathbf{y}_u, \mathbf{s}) + \lambda_{u,3} P_{u,3}(\mathbf{y}_u, \mathbf{s}) + P_{u,4}(\mathbf{y}_u, \mathbf{s})\right), \tag{30}$$

where $P_{u,i}(\mathbf{x}, \mathbf{s})$ is the IQP obtained while considering the constraint $C_{u,i}(\mathbf{x})$ in this concatenated scheme, namely as a satellite of all the constraints $C_{u,j}(\mathbf{x})$ with $j < i$ and master of those with $j > i$, for $i, j = 1, 2, 3, 4$. The tuning of the parameters $\lambda_{u,i}$ is given by the following generalization of equation (23):

$$\lambda_{u,i} = 1 + \gamma \max\left\{0, \max_{\mathbf{x},\mathbf{s}} \sum_{j>i} P_{u,j}(\mathbf{y}_u, \mathbf{s})\right\}. \tag{31}$$

Regarding the multipliers λ_u appearing in equation (30), we can adopt the same techniques discussed above in this section.

5. Case of study: MPBS

We present the locally-constrained optimization problem that we focus on in this work, which is inspired by a real-world financial scenario. Consider a network of users having a set of pending receivables that still have to be paid for. Each receivable indicates the details of a precise pending payment and therefore it is defined by its value, the corresponding creditor and debtor users and some additional temporal features, such as the dates the receivable has been submitted and when the corresponding payment falls due. These scenarios are often handled by *receivable funders* who have the role of anticipating the resolution of receivables: users are requested to pay a fee in order to have instant access to a lump sum of capital, which significantly eases the cash-flow issues associated with receivables. Nonetheless, a different strategy for receivable resolution has been proposed in [29], where a *do-ut-des* mechanism is exploited. The main goal in the design of this network-based receivable funding is to select the largest-amount subset of receivables for which autonomous payment among users is possible, without involving the funder. This optimization has to be performed under two main constraints. First, the account balance of each user has to be upper- and lower-bounded by some pre-fixed values before and after the resolution of the chosen receivables. We call this constraint CAP/FLOOR, where the CAP (FLOOR) value corresponds to the upper (lower) bound of the account balance. Secondly, each user accepts to realize at least one anticipated payment for which they are debtor if and only if they also receive at least one payment corresponding to a receivable for which they are creditor. Hence, this constraint, called IN/OUT, prevents users from only receiving or only executing the payment of receivables.

To reformulate this optimization problem in a more mathematical framework, we say that each day, we have a different set of receivables $\mathcal{R}$ that constitutes the pending transactions, namely those not resolved the day before together with those added in the meantime between the two days. Therefore, each receivable is defined by its amount $w(u, v) > 0$, the corresponding debtor u and creditor v, together with several temporal features which we do not exploit in this analysis [29]. Indeed, we focus on the single-day resolution problem defined by the R-multigraph induced by $\mathcal{R}$, formulated as:

Definition 2. (Receivables R-multigraph) *The R-multigraph induced by the set of receivables $\mathcal{R}$ consists of the quadruple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w, \mathbf{f})$, where $\mathcal{V}$ is a set of nodes, $\mathcal{E}$ is a multiset of ordered pairs of nodes called arcs, and $w : \mathcal{E} \to \mathbb{R}^+$ assigns positive numerical values to arcs and $\mathbf{f} : \mathcal{V} \to \mathbb{R}^4$ assigns four attributes to each node. The nodes $u \in \mathcal{V}$ represent the users in $\mathcal{R}$, the arcs $(u, v) \in \mathcal{E}$ represent the receivables in $\mathcal{R}$, where u represents the corresponding debtor and v the creditor, and $w(u, v)$ is the value of the receivable (u, v). Each node $u \in \mathcal{V}$ is assigned attributes $\mathbf{f}(u) = (bl_r(u), bl_a(u), cap(u), fl(u))$, respectively the corresponding receivable balance, actual balance, cap and floor values.*

Hence, given a set of receivables $\mathcal{R}$, we can identify a corresponding R-multigraph. The aim of the optimization problem that we are about to define is to maximize the overall value of the receivables paid

under the condition that the constraints CAP/FLOOR and IN/OUT described above are satisfied by all users. More precisely:

Definition 3. (Max-Profit Balanced Settlement (MPBS)) *Given the R-multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w, \mathbf{f})$ induced by the receivables $\mathcal{R}$, find*

$$\mathcal{E}^* = \arg\max_{\hat{\mathcal{E}} \subseteq \mathcal{E}} \sum_{(u,v)\in\hat{\mathcal{E}}} w(u,v)\,, \tag{32}$$

$$\text{s.t. CAP/FLOOR}\,(u),\ \forall u \in \mathcal{V}(\hat{\mathcal{E}}), \tag{33}$$

$$\text{IN/OUT(u)},\ \forall u \in \mathcal{V}(\hat{\mathcal{E}}). \tag{34}$$

where $\hat{\mathcal{E}}$ is a generic subset of all the arcs/transactions $\mathcal{E}$ and $\mathcal{V}(\hat{\mathcal{E}}) = \{u \in \mathcal{V} \,|\, (u,v) \vee (v,u) \in \hat{\mathcal{E}}\}$. Condition (33) confines the balance of the users that take part to the solution multigraph $\mathcal{E}^*$ to be inside a finite region, so that floor and cap conditions are not violated after the realization of the selected transactions:

$$\text{CAP/FLOOR}\,(u): \sum_{v:(v,u)\in\hat{\mathcal{E}}} w(v,u) - \sum_{v:(u,v)\in\hat{\mathcal{E}}} w(u,v) \in [fl(u) - bl_a(u), cap(u) - bl_r(u)]\,. \tag{35}$$

Condition (34) imposes that each user taking part to the solution multigraph $\mathcal{E}^*$ has to be debtor and creditor for at least one transaction:

$$\text{IN/OUT}\,(u): \left|\left\{(u,v) \mid (u,v) \in \hat{\mathcal{E}}\right\}\right| \geqslant 1, \text{ and } \left|\left\{(v,u) \mid (v,u) \in \hat{\mathcal{E}}\right\}\right| \geqslant 1\,. \tag{36}$$

We start by underlying the locally-constrained nature of this optimization problem: each constraint is defined node-by-node and only depends on the arcs incident to that same node. Secondly, the MPBS problem is NP-hard as it can be reduced from the SubsetSum problem [29], which is a special case of the Knapsack problem. This problem presents the same structure of other well-known minimum-cost circulation problems, which have been extensively studied for decades (see, e.g. [49]). The authors of [29, 30] exhaustively studied how to approach the MPBS problem by using classical computation resources, adopting branch-and-bound and cycle-enumeration-and-selection algorithms. Nonetheless, since MPBS belongs to the class of NP-hard problems, the exact solution of this problem cannot be found efficiently, namely in a number of steps that grows polynomially with the instance size. As we see later in this work, even finding a single feasible configuration, that is a configuration that satisfies the problem constraints, is not trivial.

In the following, we show how the IQPMS methods can be used to reduce the amount of slack variables needed for a QUBO formulation of MPBS with respect to the standard procedure of section 2.1. A simple example of this problem is depicted in figure 3.

Notice that, if a node u has either no incoming or outgoing arc in the input R-multigraph, the optimal solution of the corresponding MPBS problem cannot include any arc incident to u, otherwise IN/OUT(u) would be violated. Moreover, the exclusion of all the arcs incident to u may cause another node v to not have any incoming or outgoing arc. In the following, in order to make our exposition clearer, we assume that all the nodes of the considered input R-multigraphs have at least one incoming and one outgoing arc.

5.1. QUBO formulation of MPBS

In order to cast MPBS into a QUBO form, we proceed as follows. First, we associate to each arc $(u,v) \in \mathcal{E}$ a binary variable $x_i \in \{0,1\}$. Hence, we obtain a string of N binary variables $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, where $|\mathcal{E}| = N$ is the total number of arcs in $\mathcal{E}$. Given the one-to-one relation $(u,v) \leftrightarrow i$ we can rewrite any expression in Definition 3 with the binary variables x_i. For instance, if we use the definition $W = \text{diag}(w_1, w_2, \ldots, w_N)$, the maximization of the objective function of MPBS can be formulated as:

$$\max_{\hat{\mathcal{E}}\subseteq\mathcal{E}} \sum_{(u,v)\in\hat{\mathcal{E}}} w(u,v) = \max_{\mathbf{x}\in\{0,1\}^N} \sum_i x_i w_i = \max_{\mathbf{x}\in\{0,1\}^N} \mathbf{x}^{\mathrm{T}} W \mathbf{x} = \max_{\mathbf{x}\in\{0,1\}^N} W(\mathbf{x})\,, \tag{37}$$

where we forced the previous notation by defining: $W(\mathbf{x}) = \mathbf{x}^{\mathrm{T}} W \mathbf{x}$. From now on, we solely consider the quadratic polynomial form of QUBO elements, e.g. $W(\mathbf{x})$, instead of the corresponding matricial form, e.g. $\mathbf{x}^{\mathrm{T}} W \mathbf{x}$.

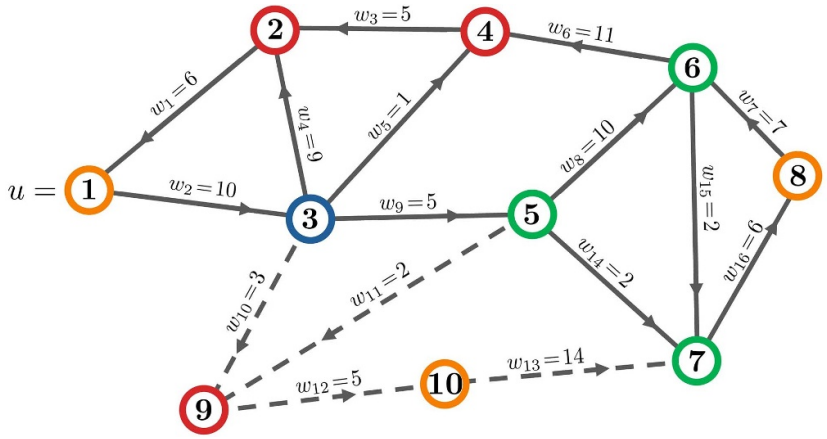


Figure 3. Instance of an R-multigraph with $|\mathcal{V}| = 10$ nodes and $N = |\mathcal{E}| = 16$ arcs, where the arrows points from the debtor to the creditor. We have nodes with $N(u) = 2$ (orange), 3 (red), 4 (green) and 5 (blue) incident arcs. The goal of MPBS is to maximize the value exchanged, namely the objective function $W(\mathbf{x}) = \sum_i x_i w_i$ where the binary variable x_i associated to the arc is equal to 0/1 if the corresponding arc is not activated/activated, under two constraints. The first, called CAP/FLOOR, requires each node to exchange a net value inside the [CAP(u),FL(u)] range, which in this case is $[-7, 8]$ for each node. For instance, if we activate all the arcs incident to the node $u = 1$, we obtain a net value of $w_1 - w_2 = -4$. The second constraint, called IN/OUT, requires each node to either activate at least one incoming and one outgoing arc or no arcs can be activated for that node. Bold arcs indicate the solution of this MPBS instance, while the activation of any dashed arc leads to a violation of CAP/FLOOR and/or IN/OUT. Indeed, node $u = 10$ cannot activate its incident arcs without violating CAP/FLOOR, as $w_{12} - w_{13} = -9$. As a consequence, the arc $(u, v) = (9, 10)$ with value $w_{12} = 5$ cannot be activated. Hence, node $u = 9$ cannot activate any of its remaining incoming arcs without violating IN/OUT. The solution subgraph $\mathcal{E}^*$ of this example is given by the combination of binary variables $\mathbf{x}^*$ such that $x_i^* = 0$ for $i = 10, 11, 12, 13$ and $x_i^* = 1$ otherwise.

Our goal is to achieve a QUBO form of MPBS defined over the N logical variables $\mathbf{x} = \{x_1, \ldots, x_N\}$ and $\overline{N}$ slack binary variables $\mathbf{s} = \{s_1, \ldots, s_{\overline{N}}\}$, hence considering the optimization of a quadratic form $Q(\mathcal{G}, \mathbf{x}, \mathbf{s})$ defined over $N + \overline{N}$ variables $\{\mathbf{x}, \mathbf{s}\}$ that corresponds to the instance of MPBS identified by $\mathcal{G}$:

$$\arg\max_{\mathbf{x},\mathbf{s}} Q(\mathcal{G}, \mathbf{x}, \mathbf{s}) \,. \tag{38}$$

Given the one-to-one relation between subsets of arcs $\hat{\mathcal{E}} \subseteq \mathcal{E}$ and binary variables combinations $\mathbf{x} \in \{0, 1\}^N$, we define:

$$\mathcal{E}^* \longleftrightarrow \mathbf{x}^* \,, \tag{39}$$

where $\mathcal{E}^*$ is the solution of MPBS. We say that the quadratic polynomial $Q(\mathcal{G}, \mathbf{x}, \mathbf{s})$ provides a QUBO formulation of MPBS if

$$\arg\max_{\mathbf{x},\mathbf{s}} Q(\mathcal{G}, \mathbf{x}, \mathbf{s}) = \{\mathbf{x}^*, \mathbf{s}^*\} \,. \tag{40}$$

Hence, in order to achieve this formulation, we must translate the constraints of MPBS, namely CAP/FLOOR and IN/OUT, through quadratic penalty terms $P^{\mathrm{CF}}(\mathbf{x}, \mathbf{s})$ and $P^{\mathrm{IO}}(\mathbf{x}, \mathbf{s})$, respectively, in order to obtain:

$$\arg\max_{\mathbf{x},\mathbf{s}} Q(\mathcal{G}, \mathbf{x}, \mathbf{s}) = \arg\max_{\mathbf{x},\mathbf{s}} W(\mathbf{x}) + P^{\mathrm{CF}}(\mathbf{x}, \mathbf{s}) + P^{\mathrm{IO}}(\mathbf{x}, \mathbf{s}) = \{\mathbf{x}^*, \mathbf{s}^*\} \,. \tag{41}$$

MPBS is locally-constrained. Indeed, constraints are clearly defined node-by-node. As we show later, we can achieve the following QUBO structure similar to equation (25):

$$Q(\mathcal{G}, \mathbf{x}, \mathbf{s}) = W(\mathbf{x}) + \sum_{u=1}^{|\mathcal{V}|} \lambda_u \left(\lambda_u^{\mathrm{IO}} P_u^{\mathrm{IO}}(\mathbf{x}, \mathbf{s}) + P_u^{\mathrm{CF}}(\mathbf{x}, \mathbf{s}) \right) , \tag{42}$$

where IN/OUT(u) is considered the master constraint of the node u, while CAP/FLOOR(u) as the corresponding satellite constraint.

We define $\mathcal{E}(u) = \mathcal{E}^+(u) \cup \mathcal{E}^-(u)$ to be the subset of arcs incident to the node u, where $\mathcal{E}^+(u)$ is the subset of incoming arcs (v, u) where u represents the creditor and $\mathcal{E}^-(u)$ is the subset of outgoing arcs (u, v) where u represents the debtor. Hence, $|\mathcal{E}(u)| = |\mathcal{E}^+(u)| + |\mathcal{E}^-(u)|$. We use the notation $N(u) = |\mathcal{E}(u)|$ and $N^{\pm}(u) = |\mathcal{E}^{\pm}(u)|$. Moreover, we use $\mathcal{E}(u)$ and $\mathcal{E}^{\pm}(u)$ to indicate specific subsets of

the binary variables $\mathbf{x}$. For instance, '$i \in \mathcal{E}^+(u)$' stands for all the values of i such that x_i corresponds to a (v,u) arc. Instead, $\overline{N}(u)$ corresponds to the amount of slack variable needed to enforce a constraint associated to the node u, where the specific constraint will be clear from the context. We remember that each constraint is enforced with independent slack variables.

We conclude this section by expressing the CAP/FLOOR and IN/OUT conditions with the binary variables x_i associated to the arcs/pending transactions $(u,v) \in \mathcal{E}$. First, it is straightforward to check that equation (35) corresponds to:

$$\text{CAP/FLOOR}(u): \; FL(u) \leqslant \sum_{i \in \mathcal{E}^+(u)} x_i w_i - \sum_{i \in \mathcal{E}^-(u)} x_i w_i \leqslant CAP(u)\,, \tag{43}$$

where $FL(u) = fl(u) - bl_a(u)$ and $CAP(u) = cap(u) - bl_r(u)$. Moreover, the IN/OUT condition for the node u given in equation (36) is equivalent to the following pair of inequality constraints:

$$\text{IN/OUT}(u): \begin{cases} \text{IC}^{\text{IO}}_{u,1}(\mathbf{x}) = -\left(|\mathcal{E}^-(u)| \sum_{i \in \mathcal{E}^+(u)} x_i\right) + \sum_{i \in \mathcal{E}^-(u)} x_i \leqslant 0\,, \\ \text{IC}^{\text{IO}}_{u,2}(\mathbf{x}) = -\left(|\mathcal{E}^+(u)| \sum_{i \in \mathcal{E}^-(u)} x_i\right) + \sum_{i \in \mathcal{E}^+(u)} x_i \leqslant 0\,. \end{cases} \tag{44}$$

5.2. Standard QUBO formulation of IN/OUT

We start by showing how the standard approach discussed in section 2.1 applies to cast the IN/OUT constraints into a QUBO form. Each IN/OUT constraint in equation (44) can be considered as a different inequality constraint $\text{IC}^{\text{IO}}_{u,j}(\mathbf{x}) \leqslant 0$, with $j = 1,2$, of a locally-constrained problem (see equation (24)).

It is easy to check that, for each u, the IN/OUT inequality constraints have the same minimum $\min_{\mathbf{x}} \text{IC}_{u,j}(\mathbf{x}) = -|\mathcal{E}^+(u)||\mathcal{E}^-(u)|$. A straight application of the standard procedure described in section 2.1 would involve the introduction of a quantity analogous to $S_j(\mathbf{s})$ (see equation (4)), where the amount $\overline{N}(u)$ of slack variables $\mathbf{s} = \{s_1, s_2, \ldots, s_{\overline{N}(u)}\}$ is chosen such that $S_j(\mathbf{s})$ can assume all the integer values in the interval $[0, |\min_{\mathbf{x}} \text{IC}^{\text{IO}}_{u,j}(\mathbf{x})|] = [0, |\mathcal{E}^+(u)||\mathcal{E}^-(u)|]$. In general, the larger is this interval, the more slack variables are necessary (see equation (6)). For instance, we could enforce $\text{IC}^{\text{IO}}_{u,1}(\mathbf{x}) \leqslant 0$ into a QUBO form through:

$$P^{\text{IO},1}_u(\mathbf{x},\mathbf{s}) = -\left(\text{IC}^{\text{IO}}_{u,1}(\mathbf{x}) + S_1(\mathbf{s})\right)^2 = -\left(-\left(|\mathcal{E}^+(u)| \sum_{i \in \mathcal{E}^-(u)} x_i\right) + \sum_{i \in \mathcal{E}^+(u)} x_i + S_1(\mathbf{s})\right)^2, \tag{45}$$

where $S_1(\mathbf{s})$ is given by equation (5) and assumes all the integer values in $[0, |\mathcal{E}^+(u)||\mathcal{E}^-(u)|]$. Nonetheless, this path would employ more slack variables than necessary.

Indeed, the combinations of $\mathbf{x}$ for which $\text{IC}^{\text{IO}}_{u,1}(\mathbf{x}) = -|\mathcal{E}^+(u)||\mathcal{E}^-(u)|$ violate $\text{IC}^{\text{IO}}_{u,2}(\mathbf{x}) \leqslant 0$ and, similarly, the combinations of $\mathbf{x}$ for which $\text{IC}^{\text{IO}}_{u,2}(\mathbf{x}) = -|\mathcal{E}^+(u)||\mathcal{E}^-(u)|$ violate $\text{IC}^{\text{IO}}_{u,1}(\mathbf{x}) \leqslant 0$. Hence, requiring $S_1(\mathbf{s})$ to be able to assume the value $|\mathcal{E}^+(u)||\mathcal{E}^-(u)|$, so that equation (45) can be zero-valued when $\text{IC}^{\text{IO}}_{u,1}(\mathbf{x}) = -|\mathcal{E}^+(u)||\mathcal{E}^-(u)|$, is not necessary, as there would be anyway a violation of the other inequality constraint $\text{IC}^{\text{IO}}_{u,2}(\mathbf{x}) \leqslant 0$. Instead, the same is not true for those $\mathbf{x}$ such that $\text{IC}^{\text{IO}}_{u,1}(\mathbf{x})$ or $\text{IC}^{\text{IO}}_{u,2}(\mathbf{x})$ are equal to the second-lowest possible value, namely $1 - |\mathcal{E}^+(u)||\mathcal{E}^-(u)|$.

As a consequence, we consider the QUBO quadratic penalty corresponding to $\text{IC}^{\text{IO}}_{u,1}(\mathbf{x}) \leqslant 0$ as equation (45), where $S_1(\mathbf{s})$ is given by equation (5) and assumes all the integer values in $[0, |\mathcal{E}^+(u)||\mathcal{E}^-(u)| - 1]$. Hence, the implementation of the first IN/OUT inequality constraint requires $\lceil \log_2(|\mathcal{E}^+(u)||\mathcal{E}^-(u)|) \rceil$ slack variables (see equation (6)).

In summary, in order to implement a quadratic penalty term that enforces IN/OUT(u) in a QUBO formulation of MPBS, we can use the standard technique described in section 2.1 and construct $P^{\text{IO}}_u(\mathbf{x},\mathbf{s}) = P^{\text{IO},1}_u(\mathbf{x},\mathbf{s}) + P^{\text{IO},2}_u(\mathbf{x},\mathbf{s})$, where $P^{\text{IO},1}_u(\mathbf{x},\mathbf{s})$ is given in equation (45) and $P^{\text{IO},2}_u(\mathbf{x},\mathbf{s})$ can be obtained similarly. We underline that, while these two addendum share the same logical binary variables, the slack variables are disjoint, namely those involved in $P^{\text{IO},1}_u(\mathbf{x},\mathbf{s})$ do not enter in the definition of $P^{\text{IO},2}_u(\mathbf{x},\mathbf{s})$ and vice-versa. The slack variable cost of this implementation for each node is:

$$\overline{N}^{\text{IO}}(u) = 2\left\lceil \log_2\left(\left|\mathcal{E}^+(u)\right| \left|\mathcal{E}^-(u)\right|\right)\right\rceil .$$

Hence, the number of slack variables needed to implement the IN/OUT constraint in the whole R-multigraph is:

$$\overline{N}^{\text{IO}} = 2 \sum_{u \in \mathcal{V}(u)} \left\lceil \log_2 \left|\mathcal{E}^+(u)\right| \left|\mathcal{E}^-(u)\right| \right\rceil . \tag{46}$$

Table 1. Amount of slack variables $\overline{N}(u)$ necessary to the standard method to enforce IN/OUT(u), where $N(u)$ is the number of arcs. The Xvs$Y$ scenario occurs when on the target node there are X arcs of one type (incoming/outgoing) and Y of the opposite type.

Arcs	Scenario	IN/OUT (standard)
$N(u)=2$	1vs1	$\overline{N}(u)=0$
$N(u)=3$	1vs2	$\overline{N}(u)=2$
$N(u)=4$	1vs3	$\overline{N}(u)=4$
	2vs2	$\overline{N}(u)=4$
$N(u)=5$	1vs4	$\overline{N}(u)=4$
	2vs3	$\overline{N}(u)=6$

In table 1 we report the amount of slack variables required by the standard method to enforce IN/OUT(u) in a QUBO.

We notice that the amount of slack variables needed for the QUBO formulation of this constraint strictly depends on the topology of the multigraph considered. Indeed, the amount of slack variables given by equation (46) is a function of the number of incoming and outgoing arcs of each node.

5.3. Standard QUBO formulation of CAP/FLOOR

We proceed by applying the standard method discussed in section 2.1 to convert CAP/FLOOR(u) into a QUBO form. This constraint can be considered as one of the linear inequality constraints in equation (2), where now $S(\mathbf{s})$ has to assume all the integer values in $[0, CAP(u)-FL(u)]$. Hence, to obtain the corresponding QUBO form, we use equation (5) to define the matrix P_u^{CF} that appears in equation (41) and we obtain:

$$P_u^{\mathrm{CF}}(\mathbf{x},\mathbf{s}) = -\left(\sum_{i\in\mathcal{E}^+(u)} w_i x_i - \sum_{i\in\mathcal{E}^-(u)} w_i x_i - CAP(u) + S(\mathbf{s})\right)^2. \tag{47}$$

Hence, the implementation of this quadratic penalty term requires $\overline{N}^{\mathrm{CF}}(u) = \lceil \log_2(1+CAP(u)-FL(u))\rceil$ slack variables for each node (see equation (6)) and therefore a total of

$$\overline{N}^{\mathrm{CF}} = \sum_{u\in\mathcal{V}} \lceil \log_2(1+CAP(u)-FL(u))\rceil \tag{48}$$

slack variables to implement the CAP/FLOOR constraint in the whole R-multigraph.

Hence, when adopting the standard method presented in section 2.1, differently from the IN/OUT case, the slack variable cost for this constraint does not depend on the topology of the graph, i.e. the amount of incoming and outgoing arcs per node, but it only depends on the values of $CAP(u)$ and $FL(u)$. Notice that, in case of multigraphs taken from real-world financial scenarios, the order of magnitude of $CAP(u)-FL(u)$ can easily be around $10^5 \sim 10^6$. Therefore, the amount of slack variables $\overline{N}^{\mathrm{CF}}(u)$ needed per node, even in case of few incoming and outgoing transactions, can be larger than 16. Later, when considering real instances of MPBS, we set $CAP(u)-FL(u)=15$ in order to need 4 slack variables per node for the enforcement of this constraint.

5.4. MPBS on quantum annealers

We can plug the newly formulated quadratic penalties for CAP/FLOOR and IN/OUT given respectively in equations (47) and (45), into equation (42) and obtain the standard QUBO formulation of MPBS. As a result, the number of binary variables over which we optimize is given by $N=|\mathcal{E}|$ variables x_i, which are associated to the arcs of the input R-multigraph, $\overline{N}^{\mathrm{IO}}$ slack variables to enforce the IN/OUT (see equation (46)) and $\overline{N}^{\mathrm{CF}}$ for CAP/FLOOR (see equation (48)).

One possibility to solve a QUBO problem with quantum technologies is via a quantum annealer, e.g. those commercially produced by D-wave [31]. In the best-case scenario, namely when each QUBO variable is not coupled with too many variables and the topology of multigraph of arcs $\mathcal{E}$ is particularly fortunate, D-wave annealers employ around one qubit per QUBO variable. Hence, by using the standard technique described above, the qubits employed to include CAP/FLOOR and IN/OUT are in general much more than those representing the actual transactions. As a result, any attempt to use classical or quantum QUBO solvers in order to solve MPBS for larger and larger R-multigraphs is highly limited by the number of slack variables required by this procedure.

Table 2. Amount of slack variables $\overline{N}(u)$ necessary to enforce IN/OUT(u) and CAP/FLOOR(u) with the IQPMS methods, where $N(u)$ is the number of arcs. IN/OUT(u) is considered as a master constraint and CAP/FLOOR(u) as its satellite. The Xvs$Y$ scenario occurs when on the target node there are X arcs of one type (incoming/outgoing) and Y of the opposite type. Comparing the slack variable cost of the IQPMS methods with the standard method, the slack variable cost of our methods is significantly lower and solely depends on the topology of the problem with graph (see table 1 and section 5 for a comparison). Indeed, differently to the case of CAP/FLOOR(u) enforcing in a QUBO form with the standard method (see equation (48)), $\overline{N}(u)$ never depends on the values of the problem parameters.

Arcs	Scenario	IN/OUT (IQPMS)	CAP/FLOOR (IQPMS)
$N(u)=2$	1vs1	$\overline{N}(u)=0$	$\overline{N}(u)=0$
$N(u)=3$	1vs2	$\overline{N}(u)=0$	$\overline{N}(u)=0$
$N(u)=4$	1vs3	$\overline{N}(u)=1$	$\overline{N}(u)=0$
	2vs2	$\overline{N}(u)=1$	$\overline{N}(u)=0$
$N(u)=5$	1vs4	$\overline{N}(u)=1$	$\overline{N}(u)=0,1$
	2vs3	$\overline{N}(u)=2$	$\overline{N}(u)=0,1,2$

While in the near future quantum processing units with more qubits and higher connectivities may be introduced and therefore better computing performances may be achieved, any reduction of the slack variables needed to implement MPBS as a QUBO makes the use of quantum annealers more suitable to solve instances of this problem. In case of more advantageous strategies, such reductions could give the opportunity to approach real-world problems with a larger input size. An approach that could be adopted together with the slack variable reduction in order to efficiently use quantum annealers is given by splitting large QUBO problems in smaller sub-QUBOs, e.g. by considering the algorithm QBSOLV [50]. This method allows to provide sub-optimal solutions to QUBO problems that would not fit entirely on a quantum chip.

In the following sections we first analyse the features of the realistic R-multigraphs studied in [30] and later we show why they are particularly suited for the application of our newly introduced techniques, namely the IQPMS methods. Real-world instances of MPBS are indeed characterized by a low number of arcs per node and therefore, as discussed in section 3.1, the adoption of IQPMS is particularly effective. Step-by-step, we show how a drastic slack variables reduction is possible. Thanks to our approach, all the nodes with 3 or less arcs require no slack variables in order for IN/OUT(u) and CAP/FLOOR(u) to be imposed in a QUBO form. Moreover, for 4-arc nodes, we need only 1 slack variable to implement IN/OUT(u). Instead, for nodes with 5 arcs, we need either 1 or 2 slack variables for IN/OUT(u) and from 0 to 2 slack variables for CAP/FLOOR(u). We summarize these slack variable costs in table 2. Hence, compared to the standard method presented in section 2.1, our methods implement significantly less slack variables and this amount depends only on the multigraph topology: no dependency from the problem parameters occurs. We remind that the standard method enforces CAP/FLOOR(u) in a QUBO form with a slack variable cost that directly depends on $CAP(u)-FL(u)$ (see equation (48)). Finally, we provide various approaches to tune the corresponding penalty multipliers.

5.5. Realistic datasets

In [29, 30] the authors show how to obtain sub-optimal solutions of MPBS for real scenarios obtained by the pan-European bank company UniCredit. The authors consider time-dependent R-multigraph, which vary day-by-day depending on the transactions resolved each day and the newly submitted ones. The average number of arcs per node obtained by averaging over a time span of two years is (see table 2 from [30]):

$$2\frac{\langle|\mathcal{E}|\rangle_{\mathrm{DATA}}}{\langle|\mathcal{V}|\rangle_{\mathrm{DATA}}}\in[2.04,\,2.77]\,, \tag{49}$$

where these averages vary depending on the particular problem parameters chosen, e.g. cap(u), fl(u) and the lifetime that each receivable is allowed to stay in the R-multigraph. Instead, the single days with the largest ratios are such that:

$$2\frac{\langle|\mathcal{E}|\rangle_{\mathrm{MAX}}}{\langle|\mathcal{V}|\rangle_{\mathrm{MAX}}}\in[2.80,\,4.59]\,. \tag{50}$$

The authors of [30] also analyse the cases where the CAP constraint is dropped (while FLOOR is still enforced), namely for cap(u)$=+\infty$. Nonetheless, the corresponding values of equations (49) and (50) do not change significantly.

It is evident that a technique that allows a reduction of the number of slack variables needed to enforce MPBS constraints for nodes with a number of arcs inside the ranges described here would be of fundamental importance in order to use QUBO solvers. As anticipated, we indeed realize such a protocol which works nicely for nodes with 5 or less arcs. All the nodes with 6 or more arcs should be treated separately and would in general need more slack variables, but the techniques introduced in this work, namely the IQPMS methods, can still be exploited.

6. IQPMS methods for MPBS

We show how the techniques introduced in this work, namely the IQPMS methods, apply to obtain a QUBO representation of MPBS. We apply the IQPMS methods, where IN/OUT(u) is enforced as master constraint and CAP/FLOOR(u) as satellite. We summarize the results of this section in table 2, where we provide the number of slack variables needed by our techniques to enforce the constraints of MPBS in the QUBO form. In appendix D we use the results of this section to build the QUBO matrices obtained with the IQPMS and the standard methods of an MPBS instance defined below in section 7.

As we show, the quadratic polynomials found by the IQPMS methods enforcing IN/OUT(u) and CAP/FLOOR(u) can be non-completely connected, namely some of the parameters $a_{ij}, \overline{a}_{kl}, b_{ik}$ defining the corresponding polynomials (7) can be null. Hence, if the quadratic forms enforcing IN/OUT(u) and CAP/FLOOR(u) do not couple the same two or more variables for one or more node u, we get a QUBO where the logical variables are not completely connected and therefore solving such a problem is in general easier than in the completely-connected case. Similarly, the same applies to the slack variables, which are not completely connected to each other and to the logical variables. Remember that the standard method in general employs several more slack variables and it completely connects all the logical and slack variables to each other (see sections 5.2 and 5.3).

A remarkable result of this section is that, apart from CAPFLOOR(u) when u has 5 incident arcs, we have a parametrized QUBO form for all the constraints. More precisely, a QUBO form of INOUT(u) can be completely evaluated before defining the MPBS instance: no additional computation is needed. Regarding CAPFLOOR(u), a parametrized solution of the linear systems defining the IQPMS methods (see section 3) is obtained, meaning that the associated QUBO representation is obtained solely by verifying whether some particular combinations of the node binary variables satisfy this constraint.

Finally, given the local nature of the problem considered, the number of computational steps required by our methods is always small compared to the problem configuration space of size $2^{N+\overline{N}}$, where N ($\overline{N}$) is the total number of arc binary variables (slack variables). The number of operations required to evaluate the QUBO form of a node constraint for an instance of MPBS with a realistic number of arcs per node (see section 5.5), is in the order of $2^{N(u)+\overline{N}(u)}$ where $N(u) \in [2,5]$ ($\overline{N}(u) \in [0,2]$) is the number of arcs incident to the node (constraint slack variables). Therefore, by considering larger MPBS instances, namely larger N, the number of nodes increase linearly with N. Hence, the global pre-processing required by IQPMS is approximately what required for an average node multiplied by the number of nodes, as these methods build the QUBO node-by-node.

6.1. Reduction of slack variables for IN/OUT

An interesting feature of this constraint is that it is homogenous among the graph, namely, modulo a relabelling of the binary variables involved in u, all the nodes having the same values of $|\mathcal{E}^+(u)|$ and $|\mathcal{E}^-(u)|$ have the same $P_u^{\mathrm{IO}}(\mathbf{x},\mathbf{s})$. Moreover, the same is true if we swap the values of $|\mathcal{E}^+(u)|$ and $|\mathcal{E}^-(u)|$: the IN/OUT penalty term for a node with $|\mathcal{E}^+(u)| = 2$ and $|\mathcal{E}^-(u)| = 1$ can be simply obtained from the IN/OUT penalty given to another node having $|\mathcal{E}^+(u)| = 1$ and $|\mathcal{E}^-(u)| = 2$. Indeed, this penalty requires each node to have either none or at least one incoming and one outgoing arc, while the values associated to each arc w_i and the attributes $CAP(u)$ and $FL(u)$ do not influence this constraint. This regularity is the main reason why we pick IN/OUT as master: once that we enforce this constraint for a given pair of $|\mathcal{E}^+(u)|$ and $|\mathcal{E}^-(u)|$, we automatically obtain the IN/OUT(u) penalty term for all the nodes with either the same $|\mathcal{E}^+(u)|$ and $|\mathcal{E}^-(u)|$ or swapped values of $|\mathcal{E}^+(u)|$ and $|\mathcal{E}^-(u)|$.

We proceed by exposing how the IQP method for master constraints applies to IN/OUT. As anticipated, this method is particularly effective when $N(u)$ is not too large and therefore we fix our attention to those nodes with $N(u) \leqslant 5$ arcs.

Nodes with 2 arcs: Consider a node with $N(u) = 2$. Without loss of generality, we associate the binary variable x_1 to the incoming arc and x_2 to the outgoing one. The IQP method provides the following

solution with no slack variables

$$P_u^{\mathrm{IO}}(\mathbf{x}) = -x_1 - x_2 + 2x_1x_2 = \begin{cases} 0 & \text{if} \quad \mathbf{x} = \{0,0\},\{1,1\} \\ -1 & \text{if} \quad \mathbf{x} = \{1,0\},\{0,1\} \end{cases}, \tag{51}$$

where $\{x_1,x_2\} = \{0,0\},\{1,1\}$ are the combinations that satisfy IN/OUT, while $\{x_1,x_2\} = \{1,0\},\{0,1\}$ violate IN/OUT.

Nodes with 3 arcs: Consider a node with $N(u) = 3$. We either have $|\mathcal{E}^+(u)| = 1$ and $|\mathcal{E}^-(u)| = 2$ or $|\mathcal{E}^+(u)| = 2$ and $|\mathcal{E}^-(u)| = 1$. Hence, we simply say that we have a 1vs2 scenario and associate x_1 to the type of arc that appears only once. More precisely, if $|\mathcal{E}^+(u)| = 1$, we associate x_1 to the incoming arc and $x_{2,3}$ to the outgoing ones. Otherwise, we associate x_1 to the outgoing arc and $x_{2,3}$ to the incoming ones when $|\mathcal{E}^-(u)| = 1$.

Interestingly, even if the free parameters in the IQP with $N = 3$ are fewer than the independent conditions that have to be applied on $P_u^{\mathrm{IO}}(\mathbf{x})$, respectively $\#a_{ij} = 7 < M(u) = 2^{N(u)} = 8$, the IQP method provides infinite many solutions with no slack variables. Remember that the standard method required 2 slack variables to enforce the same constraint (see section 5.2). Hence, we replace some inequality sub-constraints with equality sub-constraints in equation (9) in order to have a larger penalty for the string that violates IN/OUT and is likely to provide a large incentive to the objective function $\sum_i w_i x_i$ of the QUBO formulation of MPBS, namely for $\{x_1,x_2,x_3\} = \{0,1,1\}$. Hence, the following linear system of 8 equations in 7 variables

$$P_u^{\mathrm{IO}}(\mathbf{x}) = a_0 + \sum_{1\leqslant i\leqslant j\leqslant 3} a_{ij}x_ix_j \quad \text{s.t.} \quad \begin{cases} P_u^{\mathrm{IO}}(\mathbf{x}) = 0 & \text{if} \quad \mathbf{x}_u = \{0,0,0\},\{1,1,0\},\{1,0,1\},\{1,1,1\} \\ P_u^{\mathrm{IO}}(\mathbf{x}) = -1 & \text{if} \quad \mathbf{x}_u = \{1,0,0\},\{0,1,0\},\{0,0,1\} \\ P_u^{\mathrm{IO}}(\mathbf{x}) \leqslant -1 & \text{if} \quad \mathbf{x}_u = \{0,1,1\} \end{cases}, \tag{52}$$

provides the following unique solution:

$$P_u^{\mathrm{IO}}(\mathbf{x}) = -x_1 - x_2 - x_3 + 2x_1x_2 + 2x_1x_3 - x_2x_3 = \begin{cases} 0 & \text{if} \quad \mathbf{x} = \{0,0,0\},\{1,1,0\},\{1,0,1\},\{1,1,1\} \\ -1 & \text{if} \quad \mathbf{x} = \{1,0,0\},\{0,1,0\},\{0,0,1\} \\ -3 & \text{if} \quad \mathbf{x} = \{0,1,1\} \end{cases}. \tag{53}$$

Nodes with 4 arcs: Consider a node with $N(u) = 4$. In the following, the 1vs3 case refers to the $|\mathcal{E}^+(u)| = 1$ and $|\mathcal{E}^-(u)| = 3$ scenario and the $|\mathcal{E}^+(u)| = 3$ and $|\mathcal{E}^-(u)| = 1$ scenario. Instead, the 2vs2 case refers to the $|\mathcal{E}^+(u)| = 2$ and $|\mathcal{E}^-(u)| = 2$ scenario.

The IQP method needs 1 slack variable in order to enforce IN/OUT as a QUBO penalty term for this type of nodes, which we call s_1. Remember that the standard method required 4 slack variables to enforce the same constraint (see section 5.2). Before providing the explicit solutions obtained for $P_u^{\mathrm{IO}}(\mathbf{x},s_1)$, we explain the ordering chosen for the binary variables. Concerning the 1vs3 case, we associate x_1 to the type of arc that appears only once in u, while $x_{2,3,4}$ are associated to the other type of arcs [51]. Instead, for the 2vs2 case, $x_{1,2}$ are associated to the incoming arcs and $x_{3,4}$ to the outgoing ones.

Finally, the corresponding quadratic forms are:

$$\begin{aligned} \text{(1vs3)}\ P_u^{\mathrm{IO}}(\mathbf{x},s_1) = {} & -x_1 - x_2 - x_3 - x_4 + x_1x_2 + 2x_1x_3 + x_1x_4 - x_2x_4 - s_1 + s_1x_1 + s_1x_2 \\ & - s_1x_3 + s_1x_4\,, \end{aligned} \tag{54}$$

$$\text{(2vs2)}\ P_u^{\mathrm{IO}}(\mathbf{x},s_1) = -x_1 - x_2 - x_3 - x_4 - x_1x_2 - x_3x_4 + s_1 + 2s_1x_1 + 2s_1x_2 + 2s_1x_3 + 2s_1x_4\,. \tag{55}$$

Concerning the sparsity of these penalty terms, we notice that the zero-valued parameters in the solution IQPs (7) are $a_{2,3} = a_{3,4} = a_0 = 0$ in the 1vs3 scenario and $a_{1,3} = a_{1,4} = a_{2,3} = a_{2,4} = a_0 = 0$ in the 2vs2 scenario.

Nodes with 5 arcs: Consider a node with $N(u) = 5$ arcs. In the following, the 1vs4 case refers to the $|\mathcal{E}^+(u)| = 1$ and $|\mathcal{E}^-(u)| = 4$ scenario and the $|\mathcal{E}^+(u)| = 4$ and $|\mathcal{E}^-(u)| = 1$ scenario. Similarly, the 2vs3 case refers to the $|\mathcal{E}^+(u)| = 2$ and $|\mathcal{E}^-(u)| = 3$ scenario and the $|\mathcal{E}^+(u)| = 3$ and $|\mathcal{E}^-(u)| = 2$ scenario.

The IQP method applied to the 1vs4 case allows a solution with 1 slack variable, which we call s_1, while for the 2vs3 case the IQP method requires 2 slack variables, which we call s_1 and s_2. Remember that the standard method required 4 and 6 slack variables, respectively, to enforce the same constraint (see section 5.2). Before providing the explicit solutions obtained for $P_u^{\mathrm{IO}}(\mathbf{x},\mathbf{s})$, we explain the ordering chosen for the binary variables. Concerning the 1vs4 case, we associate x_1 to the type of arc that appears

only once in u, while $x_{2,3,4,5}$ are associated to the other type of arcs. Instead, for the 2vs3 case, $x_{1,2}$ are associated to the type of arcs that appears twice in u, while $x_{3,4,5}$ are associated to the other type of arcs.

Finally, the corresponding quadratic forms are:

$$\begin{aligned}(1\text{vs}4)\ P_u^{\mathrm{IO}}(\mathbf{x},s_1) = &-x_1-x_2-x_3-x_4-x_5+2x_1x_2+2x_1x_3+2x_1x_4+2x_1x_5-x_2x_3-x_2x_4-x_2x_5-x_3x_4\\ &-x_3x_5-x_4x_5-5s_1+2s_1x_2+2s_1x_3+2s_1x_4+2s_1x_5\,, \end{aligned} \tag{56}$$

$$\begin{aligned}(2\text{vs}3)\ P_u^{\mathrm{IO}}(\mathbf{x},s_1,s_2) = &-x_1-x_2-x_3-x_4-x_5-x_1x_2-x_3x_4-x_3x_5-x_4x_5-2s_1-2s_2+2s_1x_1+2s_1x_2\\ &+2s_1x_3+2s_1x_4+2s_1x_5+s_2x_3+s_2x_4+s_2x_5\,. \end{aligned} \tag{57}$$

Concerning the sparsity of these penalty terms, we notice that the zero-valued parameters in the solution IQPs (7) are $b_{11}=0$ in the 1vs4 scenario and $a_{13}=a_{14}=a_{15}=a_{23}=a_{24}=a_{25}=\bar{a}_{12}=b_{12}=b_{22}=0$ in the 2vs3 scenario.

6.2. Reduction of slack variables for CAP/FLOOR

We follow by enforcing CAP/FLOOR in the QUBO form as a satellite constraint of IN/OUT, which has already been treated as a master constraint. In this section, we construct the quadratic penalty terms $P_u^{\mathrm{CF}}(\mathbf{x},\mathbf{s})$ with the IQP method by requiring that it provides penalties if $\mathbf{x}$ satisfies IN/OUT(u) and violates CAP/FLOOR(u). Moreover, this quadratic form has to give no-penalties for at least one combination of $\mathbf{s}$ whenever $\mathbf{x}$ satisfies IN/OUT(u) and CAP/FLOOR(u). In other terms, we require $P_u^{\mathrm{CF}}(\mathbf{x},\mathbf{s})$ to penalize the combinations violating CAP/FLOOR(u) solely when they already satisfy IN/OUT(u) (see section 3.2). It follows that such a $P_u^{\mathrm{CF}}(\mathbf{x},\mathbf{s})$ could provide accidental incentives when IN/OUT(u) is violated and therefore we have to proceed as described in section 3.3. Hence, we look for a IQP (7) that satisfies the linear system (14), which now assumes the form:

$$\begin{cases} P_u^{\mathrm{CF}}(\mathbf{x},\mathbf{s}) \leqslant -1 & \text{if } \mathbf{x} \text{ satisfies IN/OUT}(u) \text{ and violates CAP/FLOOR}(u) \text{ for all } \mathbf{s} \\ P_u^{\mathrm{CF}}(\mathbf{x},\mathbf{s}) \leqslant 0 & \text{if } \mathbf{x} \text{ satisfies IN/OUT}(u) \text{ and satisfies CAP/FLOOR}(u) \text{ for all } \mathbf{s} \\ P_u^{\mathrm{CF}}(\mathbf{x},\mathbf{s}) = 0 & \text{if } \mathbf{x} \text{ satisfies IN/OUT}(u) \text{ and satisfies CAP/FLOOR}(u) \text{ for at least one } \mathbf{s} \end{cases} \tag{58}$$

Before proceeding, we briefly analyse the slack variables cost expected for IQP method in this scenario. The total number of free parameters inside a IQP grows quadratically with $N(u)+\overline{N}(u)$ (see equation (8)), where $\overline{N}(u)$ is the number of slack variables employed. In case of master constraints, the total number of combinations of $\mathbf{x}$ for which we have to enforce sub-constraints are $2^{N(u)}$. The advantage of considering CAP/FLOOR(u) a satellite constraint of IN/OUT(u) is that we can ignore those sub-constraints violating IN/OUT(u), which are $2^{N^+(u)}+2^{N^-(u)}-2$ [52]. Hence, the total number of sub-constraints that have to be imposed for CAP/FLOOR(u) are:

$$\#\mathrm{cond}(u) = 2^{N(u)} - 2^{N^+(u)} - 2^{N^-(u)} + 2\,. \tag{59}$$

As discussed in section 3.1, an indicative estimate for the minimum number of slack variables necessary to IQP method to enforce a given constraint is given by that $\overline{N}(u)$ such that the number of free parameters (see equation (8)) is larger than the number of sub-constraints (see equation (59)), namely for:

$$\#\left\{a_{ij},\bar{a}_{kl},b_{ik}\right\} = 1+\frac{\left(N(u)+\overline{N}(u)\right)^2+N(u)+\overline{N}(u)}{2} > \#\mathrm{cond}(u) = 2^{N(u)} - 2^{N^+(u)} - 2^{N^-(u)} + 2\,.$$

From the values of $\#\{a_{ij},\bar{a}_{kl},b_{ik}\}$ and $\#cond(u)$ reported in table 3, we expect that one or more slack variables are necessary to enforce CAP/FLOOR(u) in a quadratic form with the IQP method only for $N(u)\geqslant 5$. As we show later, this is indeed true. Moreover, by generating several random graphs, a major part of the 5-arc nodes studied required either zero or one slack variable and very few needed two slack variables.

In case of $N(u)=2,3,4$ we need no slack variables to enforce CAP/FLOOR(u) in the QUBO form with our methods and moreover we have a closed form for the corresponding polynomials $P_u^{\mathrm{CF}}(\mathbf{x})$. Moreover, since the linear systems (58) leave free many parameters of the IQP with no slack variables, we can insert back some sub-constraints violating IN/OUT(u) while still obtaining a solution in a closed form. In particular, for $N(u)=2$ we can add 2 conditions back, and therefore we do not need to assume IN/OUT(u) to be satisfied, for $N(u)=3$ we can add 3 conditions back and for $N(u)=4$ we can add 3 conditions back in the 1vs3 scenario and 1 in the 2vs2 scenario. Instead, for what concerns the $N(u)=5$ case, we could not find a closed form of $P_u^{\mathrm{CF}}(\mathbf{x},\mathbf{s})$. Therefore, in the numerical simulations presented below, the corresponding IQPs have been calculated node by node. In addition, while looking for

Table 3. Comparison between the number of free parameters $\#\{a_{ij}, \bar{a}_{ij}, b_k\}$ in an IQP for different numbers of arcs $N(u)$ and slack variables $\overline{N}(u)$ with the number of sub-constraints $\#\text{cond}(u)$ required by CAP/FLOOR(u) when enforced as a satellite constraint of IN/OUT(u). The minimum amount of slack variables necessary is given when $\#\{a_{ij}, \bar{a}_{ij}, b_k\} > \#\text{cond}(u)$. No slack variables are indeed necessary for $N(u) \leqslant 4$, while for $N(u) = 5$ one or two may be needed. The Xvs$Y$ scenario occurs when on the target node there are X arcs of one type (incoming/outgoing) and Y of the opposite type.

Arcs	$\#\{a_{ij}, \bar{a}_{ij}, b_k\}$	$\#\text{cond}$
$N(u) = 2$	$(\overline{N}(u) = 0)$ 4	2
$N(u) = 3$	$(\overline{N}(u) = 0)$ 7	4
$N(u) = 4$	$(\overline{N}(u) = 0)$ 11	(1vs3) 8 (2vs2) 10
$N(u) = 5$	$(\overline{N}(u) = 0)$ 16 $(\overline{N}(u) = 1)$ 22 $(\overline{N}(u) = 2)$ 29	(1vs4) 16 (2vs3) 22

these polynomials, we incentivized solutions with fewer slack variables and with sparser representative matrices.

In the following, we provide the quadratic forms $P_u^{\text{CF}}(\mathbf{x})$ that are the solutions of the linear system (58) for $N(u) = 2, 3, 4$. These penalties solely depend on the particular combinations $\mathbf{x}$ that satisfy IN/OUT(u) and violate CAP/FLOOR(u). Hence, given a node u, we simply have to check which are these combinations and make a simple substitution inside a parametrized polynomial, without the need to solve any linear system. We remember that for this constraint the standard method required slack variables for each node. Consider that, if we consider realistic scenarios as those analyzed in [30], $CAP(u) - FL(u)$ can easily be greater than 10^6.

Nodes with 2 arcs: In this case CAP/FLOOR(u) and IN/OUT(u) can be enforced at the same time with a single polynomial. It is enough to find an IQP in two variables satisfying the following linear system:

$$\begin{cases} P_u^{\text{IO,CF}}(0,0) = 0 \\ P_u^{\text{IO,CF}}(x_1,x_2) = -1 \quad \text{for} \quad \{x_1,x_2\} = \{1,0\},\{0,1\} \\ P_u^{\text{IO,CF}}(1,1) = \sigma(1,1) \end{cases}$$

where $\sigma(1,1) = 0$ if $\{x_1,x_2\} = \{1,1\}$ satisfies CAP/FLOOR(u) and $\sigma(1,1) = -1$ otherwise. Remember that $\{x_1,x_2\} = \{0,0\}$ always satisfies both CAP/FLOOR(u) and IN/OUT(u) while $\{x_1,x_2\} = \{1,0\}$ and $\{x_1,x_2\} = \{0,1\}$ always violate IN/OUT(u). If $\{x_1,x_2\} = \{1,1\}$ satisfies CAP/FLOOR(u), we obtain $P_u^{\text{IO,CF}}(x_1,x_2) = -x_1 - x_2 + 2x_1x_2$. Instead, if $\{x_1,x_2\} = \{1,1\}$ violates CAP/FLOOR, we have $P_u^{\text{IO,CF}}(x_1,x_2) = -x_1 - x_2 + x_1x_2$. Indeed, we can express the solution IQP as follows:

$$P_u^{\text{IO,CF}}(\mathbf{x}) = -x_1 - x_2 + (2 + \sigma_{11})\, x_1x_2\,. \tag{60}$$

Nodes with 3 arcs: We consider the ordering introduced in section 6.1 for which x_1 is associated to the type of arc that appears only once in u, while $x_{2,3}$ correspond to the other type of arcs. Similarly to the $N(u) = 2$ case, we express the linear system (58) as a function of the CAP/FLOOR(u) penalties/no-penalties $\sigma(x_1,x_2,x_3)$ that have to be assigned to those combinations $\{x_1,x_2,x_3\}$ satisfying IN/OUT(u). In particular, we consider the system:

$$\begin{cases} P_u^{\text{CF}}(0,0,0) = 0 \\ P_u^{\text{CF}}(x_1,x_2,x_3) = -1 \quad \text{for} \quad \{x_1,x_2,x_3\} = \{0,1,1\},\{0,0,1\},\{1,0,0\} \\ P_u^{\text{CF}}(x_1,x_2,x_3) = \sigma(x_1,x_2,x_3) \quad \text{for} \quad \{x_1,x_2,x_3\} = \{1,1,0\},\{1,0,1\},\{1,1,1\} \end{cases} \tag{61}$$

where $\sigma(x_1,x_2,x_3) = 0$ if $\{x_1,x_2,x_3\}$ satisfies CAP/FLOOR(u) and $\sigma(x_1,x_2,x_3) = -1$ otherwise. Notice that, as anticipated, we added three extra sub-constraints in order to completely determine all the free parameters of $P_u^{\text{CF}}(\mathbf{x})$. Indeed, we required penalties for $\{x_1,x_2,x_3\} = \{0,1,1\},\{0,0,1\},\{1,0,0\}$, which violate IN/OUT($u$). The solution of such system is:

$$\begin{aligned} P_u^{\text{CF}}(\mathbf{x}) = {} & -x_1 + (1 + \sigma(1,0,1) + \sigma(1,1,0) + \sigma(1,1,1))\, x_2 - x_3 + (\sigma(1,1,1) - \sigma(1,0,1))\, x_1x_2 \\ & + (2 + \sigma(1,0,1))\, x_1x_3 + (\sigma(1,1,1) - \sigma(1,0,1) - \sigma(1,1,0) - 1)\, x_2x_3\,. \end{aligned} \tag{62}$$

Nodes with 4 arcs: We consider the ordering introduced in section 6.1 for which, in the 1vs3 scenario x_1 is associated to the type of arc that appears only once in u, while $x_{2,3,4}$ correspond to the other type of

arcs. Instead, for the 2vs2 scenario we consider $x_{1,2}$ associated to the incoming arcs, while $x_{3,4}$ are associated to the outgoing ones. Similarly to the $N(u)=2$ case, we express the linear system (58) as a function of the CAP/FLOOR(u) penalties/no-penalties $\sigma(x_1,x_2,x_3,x_4)$ that have to be assigned to those combinations $\{x_1,x_2,x_3,x_4\}$ satisfying IN/OUT(u). In particular, for the 1vs3 case we consider the system:

$$\begin{cases} P_u^{\mathrm{CF}}(0,0,0,0)=0 & \\ P_u^{\mathrm{CF}}(x_1,x_2,x_3,x_4)=-1 & \text{for} \quad \{x_1,x_2,x_3,x_4\}=\{0,1,1,0\},\{0,0,1,1\},\{0,1,1,1\} \\ P_u^{\mathrm{CF}}(x_1,x_2,x_3,x_4)=\sigma(x_1,x_2,x_3,x_4) & \text{for} \quad \{x_1,x_2,x_3,x_4\}=\{1,1,0,0\},\{1,0,1,0\},\{1,1,1,0\}, \\ & \qquad \{1,0,0,1\},\{1,1,0,1\},\{1,0,1,1\},\{1,1,1,1\} \end{cases} \tag{63}$$

where $\sigma(x_1,x_2,x_3,x_4)=0$ if $\{x_1,x_2,x_3,x_4\}$ satisfies CAP/FLOOR(u) and $\sigma(x_1,x_2,x_3,x_4)=-1$ otherwise. Notice that we added three extra penalty sub-constraints for $\{x_1,x_2,x_3,x_4\}=\{0,1,1,0\}$, $\{0,0,1,1\},\{0,1,1,1\}$, which violate IN/OUT(u). The solution of such system is given in appendix B.

Instead, for the 2vs2 case we consider the system:

$$\begin{cases} P_u^{\mathrm{CF}}(0,0,0,0)=0 & \\ P_u^{\mathrm{CF}}(0,0,1,1)=-1 & \\ P_u^{\mathrm{CF}}(x_1,x_2,x_3,x_4)=\sigma(x_1,x_2,x_3,x_4) & \text{for} \quad \{x_1,x_2,x_3,x_4\}=\{1,0,1,0\},\{0,1,1,0\},\{1,1,1,0\},\{1,0,0,1\}, \\ & \qquad \{0,1,0,1\},\{1,1,0,1\},\{1,0,1,1\},\{0,1,1,1\},\{1,1,1,1\} \end{cases} \tag{64}$$

where $\sigma(x_1,x_2,x_3,x_4)$ is defined analogously. Notice that we added the extra penalty sub-constraint for $\{x_1,x_2,x_3,x_4\}=\{0,0,1,1\}$, which violates IN/OUT($u$). The solution of such system is given in appendix B.

Nodes with 5 arcs: We randomly generated multigraphs and, by using the IQP method for satellite constraints, we never needed more than 1 slack variable to enforce CAP/FLOOR(u) in the 1vs4 scenario and 2 slack variables for the 2vs3 scenario. Moreover, we often encountered instances where even less slack variables were needed, namely 0 slack variables in the 1vs4 scenario and 0 or 1 slack variables in the 2vs3 scenario. Differently from the $N(u)=2,3,4$ cases, we could not achieve a closed form for $P_u^{\mathrm{CF}}(\mathbf{x},\mathbf{s})$ parametrized by some quantities analogous to the $\sigma(\mathbf{x})$ considered above. Hence, node by node, we had to solve the linear system (58).

6.3. Tuning of MPBS penalty multipliers

We showed how to implement CAP/FLOOR(u) and IN/OUT(u) conditions with a reduced number of slack variables, if compared to the standard method, for nodes $u\in\mathcal{V}$ with $N(u)\leqslant 5$ arcs. In this section we show how to set the multipliers λ_u and λ_u^{IO} in order to be sure that the combinations $\mathbf{x}$ that solve the maximization in the QUBO encoding the MPBS problem satisfy CAP/FLOOR(u) and IN/OUT(u). In other terms, we want $Q(\mathcal{G},\mathbf{x},\mathbf{s})$ to faithfully represent MPBS, and therefore to satisfy equation (41). In the following, we use the corresponding single-node contribution:

$$Q_u(\mathcal{G},\mathbf{x},\mathbf{s})=\left(\sum_{i\in\mathcal{E}(u)} w_i x_i\right)+\lambda_u\left(\lambda_u^{\mathrm{IO}}P_u^{\mathrm{IO}}(\mathbf{x},\mathbf{s})+P_u^{\mathrm{CF}}(\mathbf{x},\mathbf{s})\right). \tag{65}$$

First, we show how to set λ_u and then we proceed with λ_u^{IO}. From now on, we assume that our QUBO formulation of MPBS is made with the IQPMS methods.

The tuning of these multipliers has to take into account the possibility of particularly unfortunate scenarios that may occur. Hence, λ_u and λ_u^{IO} have to be set large enough so that also these scenarios get the right penalty factors that allow to obtain the correct optimal result $\mathbf{x}^*$ (see equation (39)). We explained a possible approach to this problem in section 3.3, where we did not assume any particular structure for the target locally-constrained problem. By studying a particular problem, such as MPBS, it may occur that its particular features allow to set smaller multipliers. While in the following we limit to apply the formulas obtained in section 3.3, in appendix C we prove that the worst-case requirements that could be find within instances of MPBS require these same multiplier bounds.

First of all, we can set the IN/OUT(u) relative multiplier λ_u^{IO} by using equation (26). The role of the satellite master and satellite constraints are now played by IN/OUT and CAP/FLOOR, respectively. Therefore, we obtain:

$$\lambda_u^{\mathrm{IO}}=1+\gamma\max\left\{0,\max_{\mathbf{x},\mathbf{s}}P_u^{\mathrm{CF}}(\mathbf{x},\mathbf{s})\right\} \tag{66}$$

where $\gamma > 1$. For what concerns λ_u, the local, neighbor and global tunings described in section 3.3 leads, respectively, to:

$$\lambda_u^{\text{local}} = \gamma \sum_{i \in \mathcal{E}(u)} w_i = \lambda w(u) \,, \tag{67}$$

$$\lambda_u^{\text{niegh}} = \gamma \sum_{v \in \text{neigh}(u)} \sum_{i \in \mathcal{E}(v)} w_i \,, \tag{68}$$

$$\lambda_u^{\text{global}} = \gamma \sum_{i \in \mathcal{E}} w_i \,, \tag{69}$$

where $\gamma > 1$ and neigh(u) is the set of nodes sharing at least one arc with u. We defined the total amount of the transactions associated to u as $w(u) = \sum_{i \in \mathcal{E}(u)} w_i$. Instead, λ_u^{neigh} is proportional to the sum of the transactions in u and its neighboring nodes. Finally, $\lambda_u^{\text{global}}$ is node-independent and is proportional to the total amount of transactions in the graph.

Concerning the possibility to consider a global tuning of λ_u, this choice surely imposes CAP/FLOOR(u) and IN/OUT(u) for all the nodes u of the R-multigraph and therefore, if we modify $Q(\mathcal{G}, \mathbf{x}, \mathbf{s})$ accordingly to these new multipliers, we get $\{\mathbf{x}^*, \mathbf{s}^*\} = \arg(\max_{\mathbf{x},\mathbf{s}} Q(\mathcal{G}, \mathbf{x}, \mathbf{s}))$. Indeed, as soon as one node violates either CAP/FLOOR(u) or IN/OUT(u), a penalty greater (in modulo) than the value of all the transactions is applied and therefore we are ensured that CAP/FLOOR(u) or IN/OUT(u) are satisfied for all the graph nodes.

Nonetheless, this approach generates very large penalties which may compromise the quality of the solution due to the evident unbalance between the weight of the incentives and the penalties for each $\mathbf{x}$. Indeed, large unbalances between the terms of a QUBO may be problematic when using quantum annealers, e.g. D-wave. For these reasons, we believe that equations (67) and (68) are the best pick to start studying the MPBS problem within this QUBO framework.

7. MPBS simulations on D-wave quantum annealers

We tested how the standard approach to translate constraints in a quadratic form compares to the IQPMS methods when they are adopted to solve instances of MPBS with D-wave quantum annealers, which are mainly designed to solve QUBO problems. We generated several random R-multigraphs $\mathcal{G}$ with different numbers of arcs and nodes. Each $\mathcal{G}$ identified a different MPBS problem, which we solved exactly by exhaustion, namely by checking all the possible combinations of $\mathbf{x}$ in equation (2). Hence, we used the standard and the IQPMS methods to realize a QUBO formulation of each MPBS problem and solved them with two different D-wave quantum annealers, Advantage_system4.1 (Adv1) and the prototype Advantage2_prototype1.1 (Adv2), respectively with 5627 and 563 operating qubits. The corresponding connectivities, namely the maximum number of qubits interacting with each qubit, are equal to 15 for Adv1 (Pegasus topology) and 20 for Adv2 (Zephyr topology) [31]. Finally, we compared the rates of success for which these annealers were able to find the corresponding optimal solutions together with the analysis of other performance figure of merits. Since all the instances have been solved on both quantum annealers, our results allow to compare their performances. For another comparative study of D-wave annealers, see, e.g. [53].

We follow by describing the features of the R-multigraphs generated. We considered 5 different arc-settings, each one with a different number of arcs, which were $N = 10, 12, 14, 16, 18$. We made this choice because, how we see later, within this range we can appreciate how the standard method goes from being satisfactory to almost useless. For each setting, we generated 4 instances of R-multigraphs with different numbers of nodes, for a total of 20 instances. Hence, fixed a setting, or number of arcs N, the different instances had different arcs/nodes ratios, ranging from 1.5 to 2.2. The average number of arcs incident to each node was $\langle N(u) \rangle \in [3, 4.4]$. In order to generate an R-multigraph with a given number of arcs and nodes, we first randomly distributed the minimal amount of arcs that allowed each node to have at least one incoming and one outgoing arc. The remaining arcs were then distributed randomly until the desired number of arcs was achieved, where we prevented this procedure to generate nodes with $N(u) \geqslant 6$. We assigned integer values w_i to arcs which were randomly sampled from the interval $[1, 18]$. Finally, we chose $FL(u) = -7$ and $CAP(u) = 8$ for all the generated instances.

After the generation of these MPBS problems, we exactly solved them by exhaustion. Hence, for each instance we collected the corresponding optimal combination $\mathbf{x}^*$ and the optimal value $W^* = \sum_i w_i x_i^*$ (see section 5.1). Moreover, in order to estimate how restrictive the constraints of this problem are and what is the probability to randomly sample a combination of $\mathbf{x}$ that simply satisfies our constraints,

we explicitly count how many combinations satisfy the problem constrains, namely IN/OUT(u) and CAP/FLOOR(u) for all u. While the possible combinations grow exponentially with N, the amount of $\mathbf{x}$ that does not lead to a constraint violation did not grow appreciably. In fact, the fraction of these feasible combinations $\mathbf{x}$ decreases exponentially with the problem size. Moreover, the optimal value of each instance has been found for a single configuration $\mathbf{x}^*$.We remind that the class of problems given by MPBS is NP-hard: no classical algorithm is known to obtain the optimal solution of these problems efficiently (in a polynomial time with respect to the input size) [29, 30].

After this analysis, we generated the QUBO representations of all the problems both with the standard and the IQPMS methods. The tuning of the penalty multipliers was executed as described in the previous sections, where we adopted the local approach and set $\gamma = 2$. The main features of the instances generated together with the corresponding amount of feasible combinations $\mathbf{x}$ satisfying the constraints and the number of variables needed to generate the described QUBO representations can be found in table 4. We report the QUBO matrices obtained with the IQPMS and the standard methods for the MPBS instance described in the first row of this table in appendix D. Moreover, in figure 4(left) we show how the average number of QUBO variables needed changes when we increase the number of arcs N in the instances considered. We show that the standard method needs approximately 4.88 variables per arc added to the graph, while the IQPMS methods need only 1.30 variables.

The number of variables needed by the IQPMS methods are much fewer, in general less than a third. Moreover, the ratio between the variables needed with the standard and the IQPMS methods increases with $N(u)$. One could object that these results are highly influenced by our arbitrary choice of $[FL(u), CAP(u)] = [-7,8]$. This is in true, for better or for worse. Indeed, this choice caused the standard method to need 4 slack variables per node, as described in section 5.3. Nonetheless, as discussed above, in realistic scenarios $[FL(u), CAP(u)]$ could easily assume values in the order of 10^5 and therefore the standard method would need more than 15 slack variables per node only to implement CAP/FLOOR. Notice that our methods do not depend on the particular values of $FL(u)$ and $CAP(u)$, as shown in table 2, and therefore its performances would not change in realistic scenarios. Hence, our pick tried to find a compromise between the possibility to obtain QUBO formulations of MPBS with the standard method that were solvable by D-wave quantum annealers, namely $CAP(u) - FL(u)$ not too large, and the idea of being somehow faithful with the possible scenarios encountered in realistic scenarios, namely $CAP(u) - FL(u)$ not too small. Nonetheless, even if we would pick smaller interval $[FL(u), CAP(u)]$, and therefore requiring even less slack variables for the standard QUBO implementation of CAP/FLOOR, we remind that the IQPMS methods always require zero slack variables for this constraint when we have $N(u) \leqslant 4$ arcs. Moreover, the IQPMS methods need anyway less slack variables to implement the master constraint IN/OUT (compare tables 1 and 2).

We followed by submitting these QUBO problems to the D-wave quantum annealers Advantage_system4.1 and Advantage2_prototype1.1, where the latter is the latest quantum annealer prototype having enhanced performance. In the following, we simply call these two annealers Adv1 and Adv2, respectively. For each one of the 20 instances of MPBS described above, we performed 4k annealing runs with Adv1 and 4k annealing runs with Adv2. These runs were performed both for the QUBO forms obtained with the standard and the IQPMS methods. Moreover, the 4k runs dedicated to each instance, annealer and method were subdivided in 16 blocks of 250 runs. The motivation of this choice is the following. Every time we ask the D-wave online services to perform a given number of runs, they generate a particular embedding that maps our QUBO problem into an initialization of their quantum chip and all the runs are then performed with that particular embedding. Hence, our motivation to split the runs in 16 different blocks was to average out the possible advantages/disadvantages coming from particularly fortunate/unfortunate embeddings generated by D-wave. As a result, we performed a total 320k runs on D-wave quantum annealers. The annealing times have not been tuned and therefore all the runs had the same default duration of 20 μs.

We show the average number of qubits needed for these embeddings depending on the method and the annealer adopted, for different number of arcs in figure 4(right). We notice that, when considering the standard method, D-wave embeddings need approximately 10.5 qubits per arc introduced into the graph. Instead, when considering our method, we only have approximately 2 qubits per arc introduced into the graph. Several details regarding the embeddings of these problems can be found in table 5. In particular, depending on the number of arcs in our instance, annealer used and method adopted, we report the average number of qubits employed, the average number of qubits employed per QUBO variable of the instance and the corresponding average maximum chain lengths. The maximum chain length of a given embedding is the maximum number of qubits needed by D-wave to embed a single QUBO variable into the quantum chip. Indeed, since D-wave annealers have limited connectivity, which are 15 for Adv1 and 20 for Adv2, it often happens that more than one qubit is needed to embed a single

Table 4. Main features of the generated MPBS instances, labeled by the corresponding number of arcs N and nodes $|\mathcal{V}|$. For each instance, we provide the number of feasible combinations $\mathbf{x}$ that satisfy the MPBS constraints and the corresponding percentage with respect the total number of possible combinations, namely 2^N. Finally, we report the number of variables $N+\overline{N}$ (logical and slack) that were needed to obtain a QUBO representation with the standard and the IQPMS methods. Hence, the total number of slack variables employed by the standard and the IQPMS methods is obtained by subtracting from the corresponding QUBO variables column the number of arcs N of the corresponding instance. By considering all the generated instances, IQPMS needed from 82.5% (first instance with $N=10$ and $|\mathcal{V}|=5$) to 95.6% ($N=18$ and $|\mathcal{V}|=12$) less slack variables than the standard method.

Arcs	Instance	$\mathbf{x}$ satisfying MPBS constraints	QUBO variables (standard)	QUBO variables (IQPMS)
$N=10$	$\|\mathcal{V}\|=6$	18 (1.758%)	50	14
	$\|\mathcal{V}\|=6$	16 (1.563%)	50	15
	$\|\mathcal{V}\|=5$	7 (0.684%)	50	17
	$\|\mathcal{V}\|=5$	9 (0.879%)	50	15
$N=12$	$\|\mathcal{V}\|=8$	7 (0.171%)	60	15
	$\|\mathcal{V}\|=7$	35 (0.854%)	60	16
	$\|\mathcal{V}\|=7$	5 (0.122%)	60	16
	$\|\mathcal{V}\|=6$	17 (0.415%)	60	18
$N=14$	$\|\mathcal{V}\|=9$	26 (0.159%)	70	18
	$\|\mathcal{V}\|=8$	8 (0.049%)	70	20
	$\|\mathcal{V}\|=8$	7 (0.043%)	70	22
	$\|\mathcal{V}\|=7$	41 (0.250%)	68	21
$N=16$	$\|\mathcal{V}\|=10$	31 (0.047%)	80	20
	$\|\mathcal{V}\|=9$	32 (0.049%)	78	23
	$\|\mathcal{V}\|=9$	16 (0.024%)	80	23
	$\|\mathcal{V}\|=8$	38 (0.060%)	80	27
$N=18$	$\|\mathcal{V}\|=12$	32 (0.012%)	86	21
	$\|\mathcal{V}\|=11$	28 (0.011%)	90	24
	$\|\mathcal{V}\|=10$	8 (0.003%)	90	26
	$\|\mathcal{V}\|=9$	14 (0.005%)	90	28

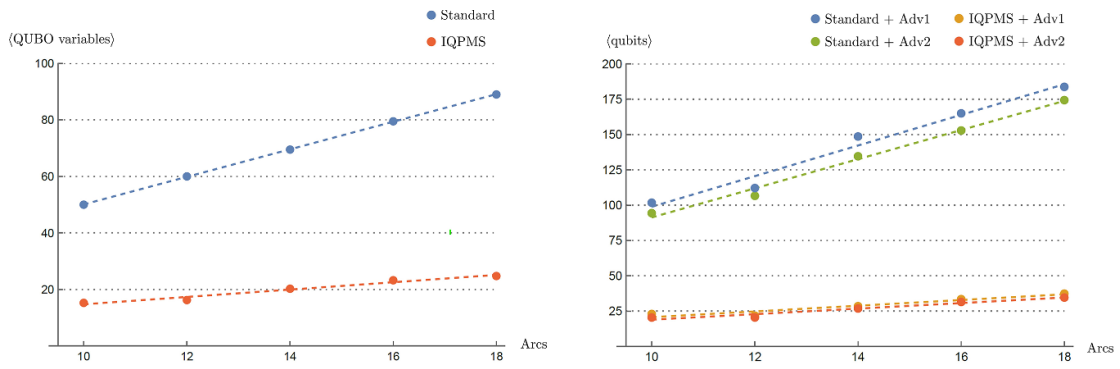


Figure 4. (Left) Average number of variables needed by the standard and IQPMS methods to provide a QUBO form of the instances generated for different number of arcs. The averages have been performed over the four instances with the same number of arcs. From the linear interpolation of these data (dashed lines), we infer that the IQPMS methods requires in average $m(\text{IQPMS})=1.30$ QUBO variables per arc added to the graph, while the standard method needs $m(\text{standard})=4.88$ QUBO variables. (Right) Average number of qubits needed to embed the QUBO instances generated with the standard and IQPMS methods into the quantum annealers Adv1 and Adv2 for different number of arcs. The averages have been performed over the four instances with the same number of arcs. From the linear interpolation of these data (dashed lines), we infer that the qubits employed per arc added to the MPBS instance to obtain an embedding on a D-wave annealer when the IQPMS and the standard methods are adopted are $m(\text{IQPMS + Adv2})=1.96$, $m(\text{IQPMS + Adv1})=2.03$, $m(\text{standard + Adv2})=10.31$ and $m(\text{standard + Adv1})=10.84$.

QUBO variable. Naturally, since the IQPMS methods require far fewer slack variables per node, if compared with the standard method, each binary variable is connected with fewer other variables and therefore our methods allow a much more efficient consumption of qubits. This is one of the main goals achieved by this work, namely providing an encouraging scaling for the qubits deployed by the embedding of locally-constrained QUBO problems into D-wave quantum annealers.

We recorded the number of successes obtained by Adv1 and Adv2 in the individuation of the MPBS optimal solutions. Moreover, we performed the same analysis by counting how many times the D-wave annealers were able to provide outputs $\mathbf{x}$ that satisfied the MPBS constraints and gave a value of the

Table 5. Several details about the D-wave embeddings of our instances of MPBS, both for the standard and the IQPMS methods. These statistics have been performed for the embeddings generated on Adv1 and Adv2 by averaging the data over all the instances of MPBS with the same number of arcs. In ⟨qubits⟩ we report the average number of qubits needed to embed our MPBS instances. In ⟨qubits/variable⟩ we report the ratio between ⟨qubits⟩ and the average number of variables needed to provide a QUBO formuation of our MPBS instances, which can be obtained from table 4. In ⟨max. chain length⟩ we report the average maximum chain length encountered in the embeddings.

Arcs	Annealer	⟨Qubits⟩ (standard)	⟨Qubits/variable⟩ (standard)	⟨Max. chain length⟩ (standard)	⟨Qubits⟩ (IQPMS)	⟨Qubits/variable⟩ (IQPMS)	⟨Max. chain length⟩ (IQPMS)
$N=10$	Adv1	101.75	2.04	5.88	23.00	1.51	2.25
	Adv2	94.38	1.89	4.63	20.50	1.34	2
$N=12$	Adv1	112.13	1.87	5.38	21.75	1.34	2.13
	Adv2	106.63	1.78	4.63	20.50	1.26	2
$N=14$	Adv1	148.63	2.14	6.38	28.50	1.41	2.38
	Adv2	134.63	1.94	6.25	26.88	1.33	2
$N=16$	Adv1	165.00	2.08	6.75	33.50	1.44	2.38
	Adv2	152.86	1.92	5.13	31.50	1.35	2.13
$N=18$	Adv1	183.75	2.06	6.63	37.38	1.51	2.63
	Adv2	174.38	1.96	5.88	34.63	1.34	2.13

Table 6. Number of annealing runs that provided either the optimal solution or an output satisfying the MPBS constraints and had an objective function within the 95% of the optimal. These data are reported for both annealers, namely Adv1 and Adv2, and the two techniques considered to obtain QUBO realizations of MPBS, namely the standard and IQPMS methods. The number of successes for each given N and annealer corresponds to the sum of the successes obtained among the 4 corresponding instances. Hence, since we performed 4k annealing runs per instance and annealer, the percentages indicated are relative to the total 16k runs performed for each arc-setting and annealer. Finally, the gain columns correspond to the ratios between the successes obtained using the IQPMS methods and the standard method. The same has been done by including those outputs satisfying the MPBS constraints and had an objective function within the 95% of the optimal.

Arcs	Annealer	Successes (standard)	Successes 95% (standard)	Successes (IQPMS)	Successes 95% (IQPMS)	Gain	Gain 95%
$N=10$	Adv1	198 (1.24%)	256 (1.60%)	1366 (8.54%)	2054 (12.84%)	6.90	8.02
	Adv2	330 (2.20%)	463 (2.89%)	3940 (23.61%)	5152 (32.20%)	11.94	11.13
$N=12$	Adv1	19 (0.12%)	36 (0.23%)	1012 (6.30%)	1149 (7.18%)	53.26	31.92
	Adv2	63 (0.39%)	85 (0.53%)	2735 (17.09%)	3149 (19.68%)	43.41	37.05
$N=14$	Adv1	6 (0.04%)	24 (0.15%)	661 (4.13%)	869 (5.43%)	110.17	36.21
	Adv2	29 (0.18%)	42 (0.26%)	1765 (11.03%)	2284 (14.28%)	60.86	54.38
$N=16$	Adv1	2 (0.01%)	8 (0.05%)	296 (1.85%)	659 (4.12%)	148.00	82.38
	Adv2	10 (0.06%)	23 (0.14%)	1149 (7.18%)	2015 (12.59%)	114.90	87.61
$N=18$	Adv1	2 (0.01%)	5 (0.03%)	322 (2.01%)	454 (2.84%)	161.00	90.80
	Adv2	4 (0.03%)	6 (0.04%)	735 (4.59%)	934 (5.84%)	183.75	155.67

objective function that was at least the 95% of the optimal value $W^* = \sum_i w_i x_i^*$, where $\mathbf{x}^*$ is the optimal solution, which are referred to as the '95% solutions'. We summarize these results in table 6, where we also include the ratio between the number of successes obtained with the IQPMS and the standard methods. These gains, both for the optimal and the 95% solutions, result to be monotonically increasing with the number of arcs in the MPBS instances. We plot the number of successes obtained in the described scenarios in figure 5.

Finally, in figure 6, we provide the performance obtained by using a classical algorithm, namely simulated annealing, to solve the same MPBS instances solved with the D-wave quantum annealers, where we show that our IQPMS QUBO formulation clearly provides higher success rates. The ratios between the successes obtained when using the two QUBO formulations, namely the gains, are: ($N=10$) 8.13, ($N=12$) 28.45, ($N=14$) 18.12, ($N=16$) 139.58, ($N=18$) 58.38, where N is the number of arcs in the instances considered. Hence, in the worst case ($N=10$) the IQPMS formulation allowed obtaining

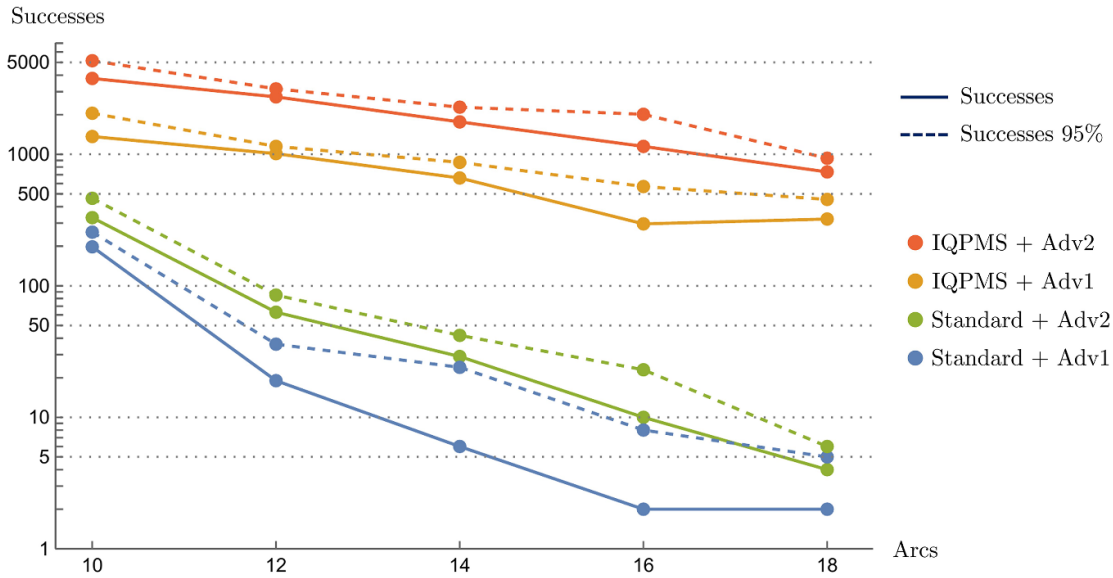


Figure 5. Number of runs, in logarithmic scale, where D-wave annealers found the optimal solutions of MPBS problems with different number of arcs after 16k runs, where their QUBO formulations were either obtained with the standard or the IQPMS methods (dots connected by solid lines). We also show the results obtained if we consider those outputs satisfying the MPBS constraints and having objective function within the 95% of the optimal (dots connected by dashed lines). These analyses were performed with both annealers Adv1 and Adv2.

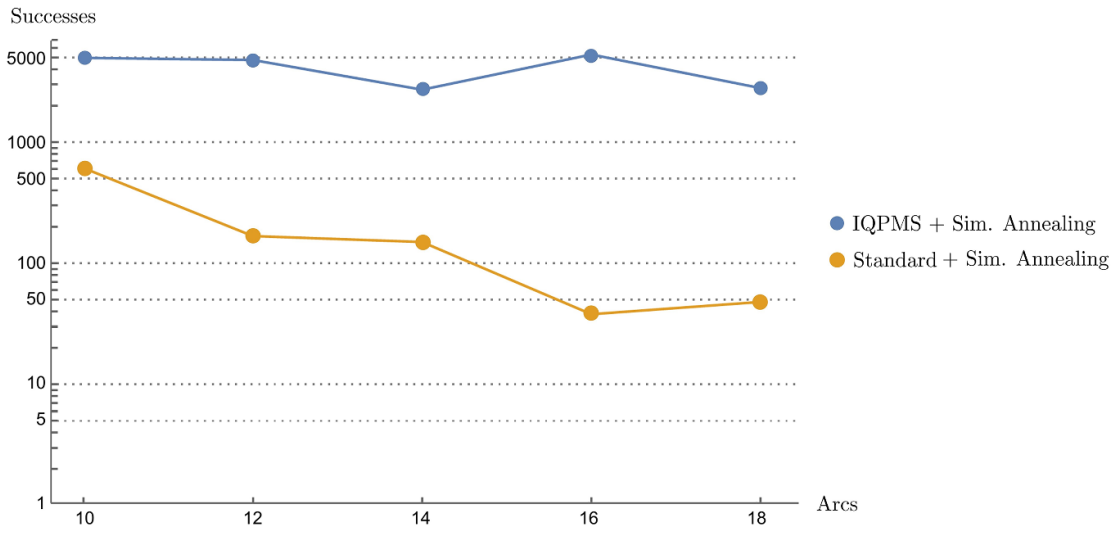


Figure 6. Number of runs, in logarithmic scale, where simulated annealing found the optimal solutions of MPBS problems with different number of arcs after 16k runs, where their QUBO formulations were either obtained with the standard (orange line) or the IQPMS methods (blue line). The MPBS instances analyzed here are the same that have been solved with D-wave quantum annealers, i.e. those analyzed in figure 5.

$\times$ 8.13 more successes than the standard approach. The trend of these gains is less obvious in this scenario as, differently from when using quantum annealing (see table 6), the decrease is not strictly monotonic. Nonetheless, it is evident that our novel methods to formulate a QUBO to solve the MPBS problem through simulated annealing have been advantageous throughout all the examined instances.

8. Discussion

We provided a novel set of techniques to translate optimization problems into a QUBO form. In particular, we showed their potential to drastically reduce the number of slack variables and to increase the sparsity of the generated QUBO problems. The first of the two techniques, the IQP method, is particularly useful whenever the constraints to be enforced are defined over few logical variables and/or the values of the parameters inside inequality constraints are large. This method does not necessary require

the locally-constrained graph problem structure. Nonetheless, in this case we can easily adopt the MS method, which provides the simultaneous enforcement of multiple constraints defined over the same set of binary variables. Moreover, we underline that our method has the same efficiency whether the constraints considered are equality/inequality linear/non-linear constraints defined with integer/non-integer parameters. Finally, we showed how to tune the corresponding constraint multipliers and we gave a generalization of the MS method for those cases where three or more constraints are defined over the same variables.

We followed by considering the NP-hard problem MPBS and we showed how to apply our methods to this real-world optimization problem from finance. Noteworthy, we obtained closed forms for the implementations of its inequality constraints INOUT(u) and CAPFLOOR(u) for all the nodes with $N(u) \leqslant 5$ arcs, with the only exception of CAPFLOOR(u) when u has 5 incident arcs, which we evaluated case by case. Hence, we achieved a parametrized QUBO form for all the other constraints, which do not need the solution of a linear system, but rather the identification of the node binary variables combinations that violate/satisfy the constraint.

We generated several instances of MPBS with different numbers of nodes and arcs. First, we quantified how better our methods work in terms of slack variables employed and later we showed how these improvements provide a much higher output quality when quantum annealers are the chosen QUBO solver. Indeed, we considered two D-wave quantum annealers, namely Advantage_system4.1 and Advantage2_prototype1.1, comparing our methods with a standard procedure that provides QUBO forms. In addition, this study provides a performance comparative study between the considered annealers.

We believe that our approach is particularly useful in the context of quantum computation as all the pre-processing applied is devoted to make the resulting QUBO as light as possible and we do not require additional rounds of computation on the (quantum) QUBO solver. Indeed, quantum annealers in general provide higher-quality outputs when problems are defined in a more compact form, namely when fewer and less-connected qubits are necessary. Hence, focusing on the refinement of QUBO formulations is clearly one key ingredient to obtain better near-term quantum computation results.

There are several research directions that could be explored starting from our results. First of all, the adoption of IQPMS to solve other combinatorial problems. Possible candidates are the MultipleKnapsack, the Multiple-ConstrainedKnapsack and the MultipleSubsetSum problems. Moreover, given the high performances obtainable in case of locally-constrained optimization problems, its employment could easily fit for real-world combinatorial problems defined over sparse networks (see, e.g. [40]).

Another interesting direction would be given by comparing our methods with those from [35]. Both output quality from quantum annealing and running times could be compared in order to understand under which regimes one could outperform the other. Indeed, while our methods may employ more slack variables, a fair comparison would allow the IQPMS methods to consider much longer quantum computation times. Moreover, it is not clear whether the number of rounds necessary to the iterative strategies are more sensitive to an increase of the input size or the number of constraints. If the latter would be the case, given realistic instances of MPBS are highly constrained, namely given by sparse graphs with node-by-node constraints, we believe that our methods should provide higher success rates. A similar comparison with the techniques presented in [40] would be fruitful as well. In summary, we expect that, if we compare the already available strategies with our novel methods in the context of solving locally-constrained lowly-connected optimization problems, we should obtain optimal solutions in shorter quantum computation times.

Future applications of the IQPMS methods could be employed with more sophisticated techniques, e.g. the tuning of the annealing times, the adoption of D-wave classical-quantum hybrid QUBO solvers or particular pre/post-processing procedures, e.g. the variable reduction proposed in [54], where an iterative algorithm fixes several binary variables to values that have a high probability of being optimal. Another path is given by the use of different NISQ devices, such as in the context of the Quantum Approximate Optimization Algorithm (QAOA) [12] or the Quantum Alternating Operator Ansatz [55], which are designed for logical-gate quantum computers. We remind that the aim of this work is to compare in the fairest way our methods with a well-established standard procedure. Hence, the setup considered has not the intent to prove how far our algorithms can be pushed to achieve optimal solutions via quantum annealing for larger and larger instances, but it rather focuses on giving a clear motivation of when and why we should adopt our novelties. Therefore, we are curious to discover up to which input size our methods, in conjunction with other techniques, allow to provide optimal solutions of MPBS instances defined via realistic datasets.

We conclude by discussing how the results presented in this work mainly support the performance of NISQ devices, while the same cannot be claimed for classical computing. The benefit deriving from

the IQPMS methods is twofold: (i) we provide a reduction of the binary (slack) variables necessary to implement optimization problems into a QUBO form and (ii) a lower connectivity among logical and slack variables is required. These two aspects allow better performances in the framework of quantum computing as the optimal quantum strategies employed to solve optimization problems make use of QUBO formulations. Indeed, feature (i), namely a reduction of the total number of binary variables considered, directly leads to fewer qubits employed in the adopted NISQ device and, as we showed in the context of quantum annealers, better performances are undoubtedly obtained. In addition, this reduction allows submitting more complex and larger problems to modern quantum computers, closer to real-world scenario optimizations. Secondly, feature (ii) backs up modern lowly-connected quantum hardware by enabling the use of fewer qubits and facilitating the minor embedding of the QUBO problem onto quantum chips. These arguments show how our contribution supports the constraints of current quantum architectures specialized in solving optimization problems, where the number of qubits employed and the required connectivity are among the main bottlenecks. We emphasize that, as the size and complexity of the problems increase, the QUBO simplification provided by our methods can become critical, enabling the successful computation of correct solutions, which would not be achievable adopting standard approaches.

A different perspective is required when considering the possible improvements of the IQPMS methods in the context of classical computation. In this case, potential improvements are not straightforward to assess, since the most effective classical approaches to the solution of constrained optimization problems do not necessarily rely on QUBO formulations. For instance, in the case of MPBS, previous works [29, 30] employed problem-specific strategies such as branch-and-bound and cycle-enumeration-and-selection algorithms, which completely bypass a QUBO encoding of the problem. As a consequence, it is unclear whether a simplification of the QUBO representation alone would translate into a tangible performance gain for classical solvers, and establishing such a connection would require a dedicated, solver-dependent analysis that goes beyond the scope of this work. By contrast, QUBO formulations constitute a central ingredient for current NISQ-era quantum approaches to optimization, including quantum annealing and gate-based algorithms such as QAOA [12]. In this context, reductions in the number of variables and in the required connectivity directly address key quantum hardware constraints, thereby enhancing the practical applicability and scalability of quantum optimization methods, in line with recent efforts aimed at enabling large-scale quantum optimization with limited qubit resources (see, e.g. [56]). Our results should therefore be interpreted as a contribution toward making more complex and structured optimization problems amenable to present-day quantum devices.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Acknowledgments

We thank Francesco Gullo and Ilaria Bordino for useful discussions. Dario De Santis acknowledges support from the research project ‘Dynamics and Information Research Institute—Quantum Information, Quantum Technologies’ within the agreement between UniCredit Bank and Scuola Normale Superiore di Pisa (CI14_UNICREDIT_MARMI). Salvatore Tirone acknowledges support from Project PRIN 2017 Taming complexity via Quantum Strategies: a Hybrid Integrated Photonic approach (QUSHIP) Id. 2017SRN-BRK and PRO3 Quantum Pathfinder. Vittorio Giovannetti acknowledges financial support by MUR (Ministero dell’Istruzione, dell’Universitá e della Ricerca) through the following projects: PNRR MUR Project PE0000023-NQSTI.

Appendix A. The MS method without the IQP method

We showed how to apply the MS method when constraints are enforced with the IQP method in section 3.2. Despite their combination, namely the IQPMS method, provides various advantages, the MS method can be adopted independently from the IQP method. Here, we show how to employ this method when constraints are enforced in a QUBO form with the standard techniques described in section 2.1. This approach could be useful when the constraints considered are defined over a number of variables that is too large to be handled by the IQP method.

Consider an optimization problem of the generic form (2). Suppose that, among all the equality and inequality constraints EC_i and IC_j, two of them are defined over the same three binary variables. For instance, consider the case where one equality and one inequality constraints are of the form:

$$\mathrm{EC}(x_1,x_2,x_3) = x_1 + x_2 + x_3 - 1\,,$$
$$\mathrm{IC}(x_1,x_2,x_3) = -2x_1 - 2x_2 + x_3 + 2\,,$$

where it is required $\mathrm{EC}(x_1,x_2,x_3) = 0$ and $\mathrm{IC}(x_1,x_2,x_3) \leqslant 0$. Our goal is to achieve quadratic forms of x_1, x_2 and x_3 that provides penalties when these constraints are violated. In the context of the MS approach, we are going to construct a penalty function corresponding to the master constraints, which provides a penalty iff it is violated, otherwise it is zero-valued. Instead, the penalty function corresponding to the satellite constraint provides penalties when (x_1,x_2,x_3) satisfy the master and violates the satellite constraint and is zero-valued when both constraints are satisfied. Hence, we do not impose any requirement to the satellite penalty function when the master constraint is violated.

We pick $\mathrm{EC}(x_1,x_2,x_3) = 0$ as master and $\mathrm{IC}(x_1,x_2,x_3) \leqslant 0$ as satellite constraint. Hence, in order to have a simultaneous reformulation of these constraints within the MS formalism, we start by enforcing $\mathrm{EC}(x_1,x_2,x_3) = 0$ with the standard method (see section 2.1), hence with the quadratic penalty function $P^{\mathrm{EC}}(x_1,x_2,x_3) = -(x_1 + x_2 + x_3 - 1)^2$.

The quadratic penalty function relative to $\mathrm{IC}(x_1,x_2,x_3) \leqslant 0$ has not to be enforced by considering all the possible values of (x_1,x_2,x_3), but solely those combinations already satisfying $\mathrm{EC}(x_1,x_2,x_3) = 0$, namely $(x_1,x_2,x_3) = (1,0,0),(0,1,0),(0,0,1)$. Notice that $(x_1,x_2,x_3) = (1,0,0),(0,1,0)$ satisfy $\mathrm{IC}(x_1,x_2,x_3) \leqslant 0$, while $(x_1,x_2,x_3) = (0,0,1)$ leads to a violation. Hence, in order to construct the quantity $S(\mathbf{s})$ needed to enforce our inequality/satellite constraint (see equation (5)), we solely have to worry to build a quadratic form $P^{\mathrm{IC}}(x_1,x_2,x_3,\mathbf{s}) = -(\mathrm{IC}(x_1,x_2,x_3) + S(\mathbf{s}))^2$ such that $\max_{\mathbf{s}} P^{\mathrm{IC}}(x_1,x_2,x_3,\mathbf{s}) = 0$ for $(x_1,x_2,x_3) = (1,0,0),(0,1,0)$ and $\max_{\mathbf{s}} P^{\mathrm{IC}}(0,0,1,\mathbf{s}) \leqslant -1$.

If $P^{\mathrm{IC}}(x_1,x_2,x_3,\mathbf{s})$ would be enforced outside the MS formalism, namely as in section 2.1, we would require $S(\mathbf{s})$ to assume all the integer values in the interval $[0,|\min_{\mathbf{x}} \mathrm{IC}(\mathbf{x})|] = [0,2]$, where the minimization is performed over all $\mathbf{x} \in \{0,1\}^3$. Hence, in the standard case we would obtain $S(\mathbf{s}) = s_0 + s_1$: two slack variables would be required. Now, since $\mathrm{IC}(x_1,x_2,x_3) \leqslant 0$ has to be enforced as a satellite of $\mathrm{EC}(x_1,x_2,x_3) \leqslant 0$, we have to ignore those strings violating the master constraint and therefore the same minimization is now performed solely over $(x_1,x_2,x_3) = (1,0,0),(0,1,0),(0,0,1)$, which leads to:

$$\left|\min_{\mathbf{x}=\{0,1\}^3} \mathrm{IC}(\mathbf{x})\right| = 2 \quad \xrightarrow{\mathrm{MS}} \quad \left|\min_{\mathbf{x}=(1,0,0),(0,1,0),(0,0,1)} \mathrm{IC}(\mathbf{x})\right| = 0.$$

Hence, by using the MS method, $\mathrm{IC}(x_1,x_2,x_3) \leqslant 0$ can be enforced with a quadratic form without employing any slack variable via $P^{\mathrm{IC}}(x_1,x_2,x_3) = -(-2x_1 - 2x_2 + x_3 + 2)^2$, which is equal to 0 iff $\mathbf{x}$ satisfies the master and the satellite constraints, namely for $(x_1,x_2,x_3) = (1,0,0),(0,1,0)$ and is smaller or equal to -1 otherwise. Hence, in this case, there are no accidental incentives and the relative multiplier of the master constraint can be fixed to $\lambda^{\mathrm{EC}} = 1$ (see section 3.3). Notice that $P^{\mathrm{IC}}(x_1,x_2,x_3)$ penalizes also those combinations of (x_1,x_2,x_3) satisfying the satellite and violating the master constraint. Hence, the sum of the two penalty functions:

$$P^{\mathrm{EC}}(x_1,x_2,x_3) + P^{\mathrm{IC}}(x_1,x_2,x_3) = -(x_1 + x_2 + x_3 - 1)^2 - (-2x_1 - 2x_2 + x_3 + 2)^2\,,$$

is equal to zero iff (x_1,x_2,x_3) satisfies both constraints and is smaller or equal to -1 iff one or both constraints are violated, where no slack variables have been employed.

Appendix B. IQPMS solutions for CAP/FLOOR(u) when $N(u) = 4$

The solution of the linear system (63) is:

$$\begin{aligned}
P_u^{\mathrm{CF}}(\mathbf{x}) = {} & (\sigma(1,0,0,1) + \sigma(1,0,1,0) - \sigma(1,0,1,1) + \sigma(1,1,0,0) - \sigma(1,1,0,1) - \sigma(1,1,1,0) \\
& + \sigma(1,1,1,1))x_1 + (2\sigma(1,0,1,1) - \sigma(1,0,0,1) - \sigma(1,0,1,0) + \sigma(1,1,0,1) + \sigma(1,1,1,0) \\
& - 2\sigma(1,1,1,1))x_2\,(1 + \sigma(1,0,1,0) - \sigma(1,0,1,1) - \sigma(1,1,1,0) + \sigma(1,1,1,1))x_3 \\
& + (-\sigma(1,0,1,0) + \sigma(1,0,1,1) - \sigma(1,1,0,0) + \sigma(1,1,0,1) + 2\sigma(1,1,1,0) - 2\sigma(1,1,1,1))x_4 \\
& + (-\sigma(1,0,1,1) + \sigma(1,1,1,1))x_1x_2 + (1 - \sigma(1,0,0,1) - \sigma(1,0,1,0) + 2\sigma(1,0,1,1) \\
& - \sigma(1,1,0,0) + \sigma(1,1,0,1) + 2\sigma(1,1,1,0) - 2\sigma(1,1,1,1))x_1x_3 + (-\sigma(1,1,1,0) \\
& + \sigma(1,1,1,1))x_1x_4 + (\sigma(1,0,0,1) - \sigma(1,0,1,1) - \sigma(1,1,0,1) + \sigma(1,1,1,1))x_2x_3
\end{aligned}$$

$$+(\sigma(1,0,1,0)-\sigma(1,0,1,1)-\sigma(1,1,1,0)+\sigma(1,1,1,1))x_2x_4+(\sigma(1,1,0,0)$$
$$-\sigma(1,1,0,1)-\sigma(1,1,1,0)+\sigma(1,1,1,1))x_3x_4\,. \tag{B1}$$

The solution of the linear system (64) is:

$$\begin{aligned}P_u^{\mathrm{CF}}(\mathbf{x}) = {} & (1-\sigma(0,1,0,1)-\sigma(0,1,1,0)+2\sigma(0,1,1,1)+\sigma(1,0,1,1)+\sigma(1,1,0,1)+\sigma(1,1,1,0)\\ & -2\sigma(1,1,1,1))x_1+(-1-\sigma(0,1,1,1)-\sigma(1,0,1,1)+\sigma(1,1,1,1))x_1x_2+(\sigma(0,1,0,1)\\ & -\sigma(0,1,1,1)-\sigma(1,1,0,1)+\sigma(1,1,1,1))x_1x_3+(\sigma(0,1,1,0)-\sigma(0,1,1,1)-\sigma(1,1,1,0)\\ & +\sigma(1,1,1,1))x_1x_4+(1+\sigma(0,1,1,1)-\sigma(1,0,0,1)-\sigma(1,0,1,0)+2\sigma(1,0,1,1)\\ & +\sigma(1,1,0,1)+\sigma(1,1,1,0)-2\sigma(1,1,1,1))x_2+(\sigma(1,0,0,1)-\sigma(1,0,1,1)-\sigma(1,1,0,1)\\ & +\sigma(1,1,1,1))x_2x_3+(\sigma(1,0,1,0)-\sigma(1,0,1,1)-\sigma(1,1,1,0)+\sigma(1,1,1,1))x_2x_4\\ & +(-1+\sigma(0,1,1,0)-\sigma(0,1,1,1)+\sigma(1,0,1,0)-\sigma(1,0,1,1)-\sigma(1,1,1,0)+\sigma(1,1,1,1))x_3\\ & +(1-\sigma(0,1,0,1)-\sigma(0,1,1,0)+2\sigma(0,1,1,1)-\sigma(1,0,0,1)-\sigma(1,0,1,0)+2\sigma(1,0,1,1)\\ & +\sigma(1,1,0,1)+\sigma(1,1,1,0)-2\sigma(1,1,1,1))x_3x_4+(-1+\sigma(0,1,0,1)-\sigma(0,1,1,1)+\sigma(1,0,0,1)\\ & -\sigma(1,0,1,1)-\sigma(1,1,0,1)+\sigma(1,1,1,1))x_4\,.\end{aligned} \tag{B2}$$

Appendix C. Worst-case scenarios for MPBS penalty multipliers

Local tuning: Consider the worst-case scenario where the optimal solution (see equation (39)) $\mathbf{x}^*$ is unique and for the node u provides the minimum contribution to the term $\sum_{i\in\mathcal{E}(u)} w_i x_i$, while there is a different combination $\mathbf{x}_{\mathrm{worst}}$ that violates CAP/FLOOR(u) and provides the maximum contribution to $\sum_{i\in\mathcal{E}(u)} w_i x_i$. Hence, we would have that the binary variables of $\mathbf{x}^*$ associated to the transactions involving u are $\{x_1^*,\ldots,x_{N(u)}^*\}=\{0,\ldots,0\}$ and $\{x_{\mathrm{worst},1},\ldots,x_{\mathrm{worst},N(u)}\}=\{1,\ldots,1\}$. We choose λ_u such that $Q_u(\mathcal{G},\mathbf{x},\mathbf{s})$ is larger for $\mathbf{x}=\mathbf{x}^*$ than for $\mathbf{x}=\mathbf{x}_{\mathrm{worst}}$:

$$\sum_{i\in\mathcal{E}(u)} w_i x_{\mathrm{worst},i}+\lambda_u P_u^{\mathrm{CF}}(\mathbf{x}_{\mathrm{worst}},\mathbf{s}) \leqslant \sum_{i\in\mathcal{E}(u)} w_i-\lambda_u \overset{!}{<} \sum_{i\in\mathcal{E}(u)} w_i x_i^*+\lambda_u P_u^{\mathrm{CF}}(\mathbf{x}^*,\mathbf{s}^*)\leqslant 0\,, \tag{C1}$$

where we used $P_u^{\mathrm{CF}}(\mathbf{x}_{\mathrm{worst}},\mathbf{s})\leqslant -1$ because $\{x_{\mathrm{worst},1},\ldots,x_{\mathrm{worst},N(u)}\}=\{1,\ldots,1\}$ satisfies the node constraints, $\{x_1^*,\ldots,x_{N(u)}^*\}=\{0,\ldots,0\}$ and the fact that $P_u^{\mathrm{CF}}(\mathbf{x}^*,\mathbf{s}^*)\leqslant 0$ because $\{x_1^*,\ldots,x_N^*\}$ satisfies CAP/FLOOR(u) and IN/OUT(u). The exclamation mark above the inequality symbol underlines that, if satisfied, λ_u can be considered large enough in this local setting.

From equation (C1), we derive the lower-bound $\lambda_u > w(u)$, where $w(u)=\sum_{i\in\mathcal{E}(u)} w_i$ is the total amount of the transactions involved with the node u. Notice that this is the same result obtained in equation (67). Hence, we fix the notation

$$\lambda_u^{\mathrm{local}}=\gamma\, w(u) \quad \text{where} \quad \gamma>1\,. \tag{C2}$$

An estimate for λ_u^{IO} can be obtained similarly. The worst-case scenario is obtained when the restriction of $\mathbf{x}^*$ on u is $\{0,\ldots,0\}$, namely providing a null contribution to $\sum_i w_i x_i$, while $\mathbf{x}_{\mathrm{worst}}$ is the combination providing the largest contribution to $\sum_i w_i x_i$ among those violating IN/OUT(u) and $P^{\mathrm{CF}}(\mathbf{x}_{\mathrm{worst}},\mathbf{s})$ provides the largest accidental incentive for $\mathbf{x}_{\mathrm{worst}}$, namely $P_u^{\mathrm{CF}}(\mathbf{x}_{\mathrm{worst}},\mathbf{s})=\max_{\mathbf{x},\mathbf{s}} P_u^{\mathrm{CF}}(\mathbf{x},\mathbf{s})=P_{\mathrm{max},u}^{\mathrm{CF}}\geqslant 0$. We choose λ_u^{IO} such that the term $\max_{\mathbf{s}} Q_u(\mathcal{G},\mathbf{x},\mathbf{s})$ is larger for $\mathbf{x}=\mathbf{x}^*$ than for $\mathbf{x}=\mathbf{x}_{\mathrm{worst}}$:

$$\begin{aligned}\max_{\mathbf{s}} Q_u(\mathcal{G},\mathbf{x}_{\mathrm{worst}},\mathbf{s}) &= \max_{\mathbf{s}}\left(\sum_{i\in\mathcal{E}(u)} w_i x_{\mathrm{worst},i}+\lambda_u\left(P_u^{\mathrm{CF}}(\mathbf{x}_{\mathrm{worst}},\mathbf{s})+\lambda_u^{\mathrm{IO}}P_u^{\mathrm{IO}}(\mathbf{x}_{\mathrm{worst}},\mathbf{s})\right)\right)\\ &< \left(1+\gamma P_{\mathrm{max}}^{\mathrm{CF}}(u)\right)w(u)-\lambda^{\mathrm{IO}}(u) \overset{!}{<} \max_{\mathbf{s}} Q_u(\mathcal{G},\mathbf{x}^*,\mathbf{s})\\ &= \max_{\mathbf{s}}\left(\sum_{i\in\mathcal{E}(u)} w_i x_i^*+\lambda_u\left(P_u^{\mathrm{CF}}(\mathbf{x}^*,\mathbf{s})+\lambda_u^{\mathrm{IO}}P_u^{\mathrm{IO}}(\mathbf{x}^*,\mathbf{s})\right)\right)=0\,.\end{aligned}$$

The first inequality is justified because $\{1,\ldots,1\}$ satisfies IN/OUT(u) and therefore $\sum_{i\in\mathcal{E}(u)} w_i x_{\mathrm{worst},i}< w(u)$, $\lambda_u=\gamma\, w(u)$, $P_u^{\mathrm{CF}}(\mathbf{x}_{\mathrm{worst}},\mathbf{s})=P_{\mathrm{max},u}^{\mathrm{CF}}$ and $P_u^{\mathrm{IO}}(\mathbf{x},\mathbf{s})\leqslant -1$ for combinations violating IN/OUT(u). Moreover, we used $\sum_i w_i x_i^*=0$ and the fact that $\mathbf{x}^*$ satisfies the node constraints. A consequence of this first inequality is that we are imposing a looser condition than previously declared [57]. Hence, we

have the lower-bound $\lambda_u^{\mathrm{IO}} > (1+\lambda P_{\max,u}^{\mathrm{CF}})/\gamma$. Since we assumed $P_{\max,u}^{\mathrm{CF}} \geqslant 0$, for the general case we can consider:

$$\lambda_u^{\mathrm{IO}} = 1+\gamma \max\left\{0, \max_{\mathbf{x},\mathbf{s}} P_u^{\mathrm{CF}}(\mathbf{x},\mathbf{s})\right\}, \text{ where } \gamma > 1. \tag{C3}$$

Non-local tuning: We set some lower-bounds (C2) and (C3) for, respectively, λ_u and λ_u^{IO} by imposing that the sub-QUBO that corresponds to the node u assumes the largest value when $\mathbf{x}$ satisfies CAP/FLOOR(u) and IN/OUT(u). We called it local because we are not considering the possible consequences that a violation of CAP/FLOOR(u) and/or IN/OUT(u) may have on $\max_{\mathbf{s}} Q_v(\mathcal{G},\mathbf{x},\mathbf{s})$ for some node $v \neq u$. Indeed, the possible incentives that these situation may generate could require to set higher values of λ_u and λ_u^{IO}.

Imagine the following unfortunate scenario: $\mathbf{x}^*$, in order to respect CAP/FLOOR(u) and IN/OUT(u), has many 0s for the variables x_i corresponding to the transactions in u. As a consequence, the transactions of the nodes v neighboring u are not activated too (otherwise CAP/FLOOR(v) or IN/OUT(v) may be violated), resulting in many other $x_i = 0$ for the arcs belonging to the nodes v sharing at least one arc with u. Hence, it may happen that a violation of CAP/FLOOR(u) and/or IN/OUT(u) in u allows the activation of many other transactions for the nodes neighboring u, where instead there are no violations of CAP/FLOOR(v) and IN/OUT(v). Therefore, there may exists a string that receives a penalty for the violation of CAP/FLOOR(u) and/or IN/OUT(u) in u, but this penalty is overcompensated by the incentives activated by the neighboring nodes (without penalties). We define the nodes adjacent to u as follows:

$$\mathrm{neigh}(u) = \{v \in \mathcal{V} \,|\, \exists (u,v) \vee (v,u) \in \mathcal{E}\}. \tag{C4}$$

Therefore, to prevent that $\arg(\max_{\mathbf{x},\mathbf{s}} Q(\mathcal{G},\mathbf{x},\mathbf{s}))$ violates CAP/FLOOR(u) and/or IN/OUT(u) as described, we consider:

$$\max_{\mathbf{x},\mathbf{s}} Q(\mathcal{G},\mathbf{x},\mathbf{s}) = \max_{\mathbf{x},\mathbf{s}} \sum_i w_i x_i + \gamma \sum_{u\in\mathcal{V}} w^{\mathrm{neigh}}(u) \left(P_u^{\mathrm{CF}}(\mathbf{x},\mathbf{s}) + \left(1+\gamma \max\left\{0, \max_{\mathbf{x},\mathbf{s}} P_u^{\mathrm{CF}}(\mathbf{x},\mathbf{s})\right\}\right) P_u^{\mathrm{IO}}(\mathbf{x},\mathbf{s}) \right), \tag{C5}$$

$$w^{\mathrm{neigh}}(u) = \sum_{v\in\mathrm{neigh}(u)} w(v),$$

where $\gamma > 1$ and is the sum of all the transactions involving u and the neighboring nodes. Notice that the multipliers involved in equation (C5) are obtained by replacing $w(u)$ with $w^{\mathrm{neigh}}(u)$ from the local formulation, namely we have:

$$\lambda_u = \gamma\, w^{\mathrm{neigh}}(u), \text{ where } \gamma > 1. \tag{C6}$$

Global tuning: In order to get a global tuning, we consider the following choice of λ_u:

$$\lambda_u = \gamma \sum_i w_i, \text{ where } \gamma > 1. \tag{C7}$$

and $\sum_i w_i$ is the sum of all the transaction defined on $\mathcal{E}$.

Appendix D. Explicit QUBO form of an instance of MPBS

In this section consider the MPBS instance described in the first line of table 4 and we give the corresponding QUBO forms obtained with the IQPMS and the standard methods. We underline that this is the smallest instance that has been considered. This example is characterized by $|\mathcal{V}| = 6$ (number of users or nodes), and $N = 10$ (total number of transactions or arcs). We can represent the users having a pending receivable, namely the nodes sharing an arc, via a correspondence matrix $w(u,v)$. More precisely

$w(u,v)=0$ when the users u and v do not have any connecting arcs, while $w(u,v)>0$ is the value of the receivable between u and v, where u is the creditor and v the debtor. Therefore, $w(u,v)$ is in general not symmetric. The correspondence matrix associated to this instance is:

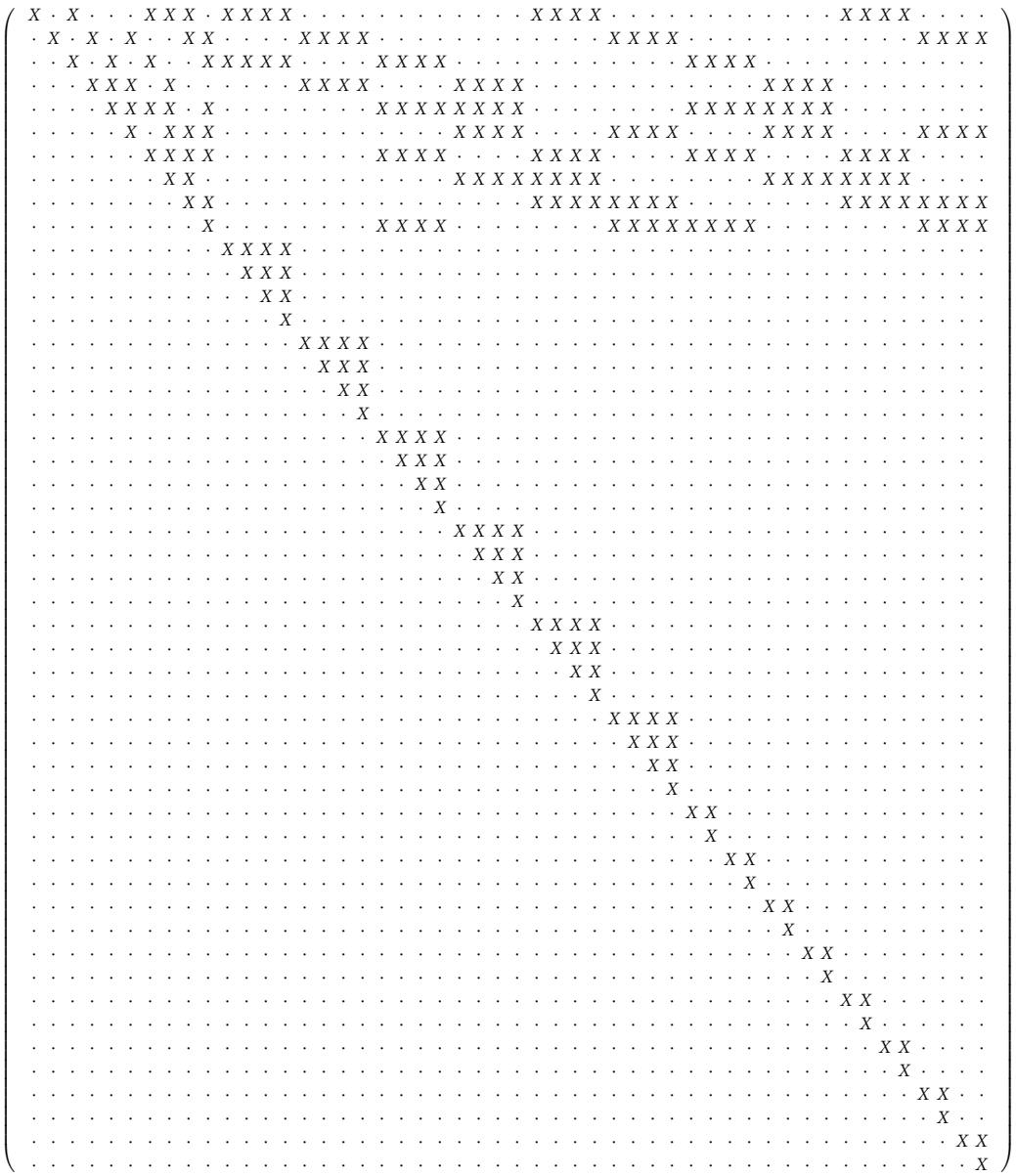
$$w(u,v)=\begin{pmatrix} 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 9 \\ 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 9 & 10 & 0 & 0 & 1 \\ 0 & 0 & 4 & 4 & 0 & 2 \\ 0 & 0 & 7 & 0 & 0 & 0 \end{pmatrix}. \tag{D1}$$

In this case, $N(u)=2$ for $u=1$, where this first user is debtor with the user $v=5$ for an amount of $w(1,5)=2$, while it is creditor with the user $v=3$ for $w(3,1)=10$. We enumerate the arcs row-by-row, meaning that x_1 is associated to the arc $(u,v)=(1,5)$, x_2 to the arc $(u,v)=(2,6)$, x_3 to the arc $(u,v)=(3,1)$ and so on. Hence, a possible configuration of the activated arcs is given by a vector of binary variables $\mathbf{x}=(x_1,x_2,\ldots,x_{10})$. It is possible to verify that, for [CAP(u),FL(u)] = [−7,8], the optimal solution of this problem (satisfying CAP/FLOOR and IN/OUT) is $\mathbf{x}^*=[1,1,1,1,0,1,1,1,0,1]$.

We obtained the following QUBO matrix using the IQPMS methods, acting on 14 variables, where the first 10 are associated to the instance arcs, while the remaining 4 correspond to the slack variables needed to implement the problem constraints:

$$\begin{pmatrix} -142 & 0 & 48 & 0 & 0 & 0 & 72 & 264 & 72 & 0 & 0 & 0 & 96 & 0 \\ 0 & -65 & 0 & 72 & 0 & -38 & 0 & 0 & 0 & 38 & 0 & 0 & 0 & 38 \\ 0 & 0 & -324 & 0 & 186 & 0 & 682 & 0 & 0 & 186 & 248 & 0 & 0 & 0 \\ 0 & 0 & 0 & -75 & -144 & 0 & 0 & 48 & 0 & 0 & 0 & 96 & 0 & 0 \\ 0 & 0 & 0 & 0 & -162 & 0 & -62 & 48 & 0 & -310 & 248 & 96 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -229 & 0 & 288 & 0 & 38 & 0 & -96 & 0 & 38 \\ 0 & 0 & 0 & 0 & 0 & 0 & -416 & -24 & -120 & -62 & -248 & 0 & 96 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -236 & -24 & 0 & 0 & 96 & -96 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -122 & 114 & 0 & 0 & 96 & -38 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -155 & 248 & 0 & 0 & 38 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -248 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -96 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -96 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -38 \end{pmatrix}. \tag{D2}$$

We report the QUBO matrix of the same instance obtained with the standard approach. This QUBO is defined over 50 binary variables, where the first 10 are associated to the arcs, while the last 40 are slack variables. For the sake of space, each zero component has been reported with a '·' symbol, while any non-zero element is marked with an '*X*':



ORCID iDs

Dario De Santis 0000-0001-5666-1770
Salvatore Tirone 0000-0002-4880-4329
Stefano Marmi 0000-0001-5105-8876
Vittorio Giovannetti 0000-0002-7636-9002

References

[1] Hauke P, Katzgraber H G, Lechner W, Nishimori H and Oliver W D 2020 Perspectives of quantum annealing: methods and implementations *Rep. Prog. Phys.* **83** 054401
[2] Yarkoni S, Raponi E, Bäck T and Schmitt S 2022 Quantum annealing for industry applications: introduction and review *Rep. Prog. Phys.* **85** 104001
[3] Abbas A *et al* 2024 Challenges and opportunities in quantum optimization *Nat. Rev. Phys.* **6** 718–35
[4] Karp R M 1972 Reducibility among combinatorial problems *Complexity of Computer Computations: Proc. of a Symposium on the Complexity of Computer Computations* (Springer) pp 85–103
[5] Lucas A 2014 Ising formulations of many np problems *Front. Phys.* **2** 5
[6] Rajak A, Suzuki S, Dutta A and Chakrabarti B K 2023 Quantum annealing: an overview *Phil. Trans. R. Soc. A* **381** 17
[7] Das A and Chakrabarti B K 2008 Colloquium: quantum annealing and analog quantum computation *Rev. Mod. Phys.* **80** 1061
[8] Farhi E, Goldstone J, Gutmann S, Lapan J, Lundgren A and Preda D 2001 A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem *Science* **292** 472
[9] Farhi E, Goldstone J and Gutmann S 2014 A quantum approximate optimization algorithm (arXiv:1411.4028 [quant-ph])
[10] Khumalo M T, Chieza H A, Prag K and Woolway M 2022 An investigation of ibm quantum computing device performance on combinatorial optimisation problems *Neural Computing and Applications* vol 1

[11] Chieza H A, Khumalo M T, Prag K and Woolway M 2020 On the computational performance of ibm quantum devices applied to combinatorial optimisation problems *2020 7th Int. Conf. on Soft Computing & Machine Intelligence (ISCMI)* pp 260–4
[12] Zhou L, Wang S-T, Choi S, Pichler H and Lukin M D 2020 Quantum approximate optimization algorithm: performance, mechanism and implementation on near-term devices *Phys. Rev. X* **10** 021067
[13] Ebadi S *et al* 2022 Quantum optimization of maximum independent set using rydberg atom arrays *Science* **376** 1209
[14] Nguyen M-T, Liu J-G, Wurtz J, Lukin M D, Wang S-T and Pichler H 2023 Quantum optimization with arbitrary connectivity using rydberg atom arrays *PRX Quantum* **4** 010316
[15] Cain M, Chattopadhyay S, Liu J G, Samajdar R, Pichler H and Lukin M D 2023 Quantum speedup for combinatorial optimization with flat energy landscapes (arXiv:2306.13123 [quant-ph])
[16] Altshuler B, Krovi H and Roland J 2010 Anderson localization makes adiabatic quantum optimization fail *Proc. Natl Acad. Sci.* **107** 12446
[17] Dickson N G and Amin M H S 2011 Does adiabatic quantum optimization fail for np-complete problems? *Phys. Rev. Lett.* **106** 050502
[18] Orús R, Mugel S and Lizaso E 2019 Quantum computing for finance: overview and prospects *Rev. Phys.* **4** 100028
[19] Mugel S, Kuchkovsky C, Sánchez E, Fernández-Lorenzo S, Luis-Hita J, Lizaso E and Orús R 2022 Dynamic portfolio optimization with real datasets using quantum processors and quantum-inspired tensor networks *Phys. Rev. Res.* **4** 013006
[20] Venturelli D and Kondratyev A 2019 Reverse quantum annealing approach to portfolio optimization problems *Quantum Mach. Intell.* **1** 17
[21] Herman D, Googin C, Liu X, Galda A, Safro I, Sun Y, Pistoia M and Alexeev Y 2022 A survey of quantum computing for finance (arXiv:2201.02773 [quant-ph])
[22] Buonaiuto G, Gargiulo F, De Pietro G, Esposito M and Pota M 2023 Best practices for portfolio optimization by quantum computing, experimented on real quantum devices *Sci. Rep.* **13** 19434
[23] Egger D J, Gambella C, Marecek J, McFaddin S, Mevissen M, Raymond R, Simonetto A, Woerner S and Yndurain E 2020 Quantum computing for finance: state-of-the-art and future prospects *IEEE Trans. Quantum Eng.* **1** 1–24
[24] Weinberg S J, Sanches F, Ide T, Kamiya K and Correll R 2023 Supply chain logistics with quantum and classical annealing algorithms *Sci. Rep.* **13** 4770
[25] Ariño Sales J F and Araos R A P 2023 Adiabatic quantum computing impact on transport optimization in the last-mile scenario *Front. Comput. Sci.* **5** 564
[26] Hernández F, Díaz K, Forets M and Sotelo R 2020 Application of quantum optimization techniques (qubo method) to cargo logistics on ships and airplanes *2020 IEEE Congreso Bienal de Argentina (ARGENCON)* pp 1–1
[27] Zinner M, Dahlhausen F, Boehme P, Ehlers J, Bieske L and Fehring L 2021 Quantum computing's potential for drug discovery: early stage industry dynamics *Drug Discovery Today* **26** 1680
[28] Glover F, Kochenberger G, Hennig R and Du Y 2022 Quantum bridge analytics I: a tutorial on formulating and using qubo models *Ann. Oper. Res.* **314** 141
[29] Bordino I and Gullo F 2018 Network-based receivable financing *Proc. 27th ACM Int. Conf. on Information and Knowledge Management* (*CIKM'18*) (ACM)
[30] Bordino I, Gullo F and Legnaro G 2020 The power of connection: leveraging network analysis to advance receivable financing (arXiv:2006.13738 [cs.DS])
[31] D. S. Inc. D-wave technical report: technical description of the D-wave quantum processing unit *Technical Report*
[32] Boyd S, Parikh N, Chu E, Peleato B and Eckstein J 2011 Distributed optimization and statistical learning via the alternating direction method of multipliers *Found. Trends Mach. Learn.* **3** 1
[33] Hestenes M R 1969 Multiplier and gradient methods *J. Optim. Theory Appl.* **4** 303
[34] Powell M J 1969 A method for nonlinear constraints in minimization problems *Optimization* p 283
[35] Djidjev H N 2023 Quantum annealing with inequality constraints: the set cover problem *Adv. Quantum Technol.* **6** 2300104
[36] Yonaga K, Miyama M J and Ohzeki M 2020 Solving inequality-constrained binary optimization problems on quantum annealer (arXiv:2012.06119 [quant-ph])
[37] Mücke T and Sascha G 2023 Efficient light source placement using quantum computing *Lernen, Wissen, Daten, Analysen (LWDA) Conf. Proc.*
[38] Verma A and Lewis M 2021 Variable reduction for quadratic unconstrained binary optimization (arXiv:2105.07032 [math.OC])
[39] van der Linde S G, van der Schoot W and Phillipson F 2023 Efficient quantum solution for the constrained tactical capacity problem for distributed electricity generation *Innovations for Community Services* ed U R Krieger, G Eichler, C Erfurth and G Fahrnberger (Springer) pp 203–21
[40] Colucci G, Linde S v d and Phillipson F 2023 Power network optimization: a quantum approach *IEEE Access* **11** 98926
[41] Kirkpatrick S, Gelatt C D and Vecchi M P 1983 Optimization by simulated annealing *Science* **220** 671
[42] van Laarhoven P J M and Aarts E H L 1987 Simulated annealing *Simulated Annealing: Theory and Applications* (Springer) pp 7–15
[43] Wilhelm M R and Ward T L 1987 Solving quadratic assignment problems by simulated annealing *IIE Trans.* **19** 107
[44] Wang C, Chen H and Jonckheere E 2016 Quantum versus simulated annealing in wireless interference network optimization *Sci. Rep.* **6** 25797
[45] Paul G 2010 Comparative performance of tabu search and simulated annealing heuristics for the quadratic assignment problem *Oper. Res. Lett.* **38** 577
[46] If $\mathrm{IC}_1(\mathbf{y}) \leqslant 0$ and $\mathrm{IC}_2(\mathbf{y}') \leqslant 0$ are enforced by $P_1(\mathbf{y},\mathbf{s})$ and $P_2(\mathbf{y}',\mathbf{s}')$, respectively, they would make use of $\overline{M}$ and $\overline{M}'$ independent slack variables
[47] Extending our approach to node-dependent $m_{IC,u}$ and $m_{EC,u}$ is straightforward but, for the sake of simplicity, we consider each node having the same number of equality and inequality constraints
[48] If equation (2) is locally-constrained, it can be written in the form of equation (24) via the rescaling $m_{EC/IC} \to m_{EC/IC}/|\mathcal{V}|$.
[49] Goldberg A V and Tarjan R E 1990 Finding minimum-cost circulations by successive approximation *Math. Oper. Res.* **15** 430
[50] D. S. Inc. 2017 D-wave technical report: partitioning optimization problems for hybrid classical/quantum execution *Technical Report*
[51] For instance, if $|\mathcal{E}^+(u)| = 3$ and $|\mathcal{E}^-(u)| = 1$, we associate x_1 to the only outgoing arc of the node
[52] The number of combinations of $\mathbf{x}$ such that there are no outgoing (incoming) arcs activated are $2^{N^+(u)}$. The number of combinations of $\mathbf{x}$ such that there are no incoming arcs activated are $2^{N^-(u)}$. Since the null combination $\mathbf{x} = \{0,0,\ldots,0\}$ is allowed by IN/OUT(u), we obtain that the number of combinations violating IN/OUT(u) are $2^{N^+(u)} + 2^{N^-(u)} - 2$

[53] Hahn G, Pelofske E and Djidjev H N 2023 Posiform planting: generating qubo instances for benchmarking *Front. Comput. Sci.* **5** 948
[54] Karimi H, Rosenberg G and Katzgraber H G 2017 Effective optimization using sample persistence: a case study on quantum annealers and various monte carlo optimization methods *Phys. Rev. E* **96** 043312
[55] Hadfield S, Wang Z, O'Gorman B, Rieffel E G, Venturelli D and Biswas R 2019 From the quantum approximate optimization algorithm to a quantum alternating operator ansatz *Algorithms* **12** 34
[56] Sciorilli M, Borges L, Patti T L, García-Martín D, Camilo G, Anandkumar A and Aolita L 2025 Towards large-scale quantum optimization solvers with few qubits *Nat. Commun.* **16** 476
[57] A tighter version of $\lambda^{IO}(u)$ could be formulated, but we do not see any particular advantage in performing the extra computation required