Université de Lille

Thèse de doctorat

Quantum Computing for Operations Research: A special approach for Project Scheduling

Thèse de doctorat de l'Université de Lille
préparée à IESEG School of Management

École doctorale n°73 Sciences Economiques, Sociales, de l'Aménagement et du Management (SESAM)
Spécialité de doctorat: Recherche Opérationnelle (OR)

Thèse présentée et soutenue à IESEG School of Management, Lille, FR, le 16-04-2024, par

LUIS FERNANDO PÉREZ ARMAS

Composition du Jury :

Jeroen BELIEN PhD, Professor., HDR, KU Leuven	Rapporteur
Riad AGGOUNE PhD, Associate Professor., HDR, Luxembourg Institute of Science and Technology	Rapporteur
Emmanuel HAVEN BA, MA, PhD, Professor., Memorial University of Newfoundland	Examinateur
Kristiaan KERSTENS PhD, Research Director., HDR, CNRS-IESEG	Jury President
Christine DI MARTINELLY PhD, Associate Professor.,HDR, IESEG	Examinateur
Stefan CREEMERS PhD, Professor., HDR, IESEG	Directeur de thèse
Samuel DELEPLANQUE PhD, Associate Professor., JUNIA-ISEN	Invité
Tanja MLINAR PhD, Associate Professor., IESEG	Invité

Université de Lille

Thèse de doctorat

Informatique quantique pour la recherche opérationnelle : Une approche spéciale pour Ordonnancement de projet

Thèse de doctorat de l'Université de Lille
préparée à IESEG School of Management

École doctorale n°73 Sciences Economiques, Sociales, de l'Aménagement et du Management (SESAM)
Spécialité de doctorat: Recherche Opérationnelle (OR)

Thèse présentée et soutenue à IESEG School of Management, Lille, FR, le 16-04-2024, par

LUIS FERNANDO PÉREZ ARMAS

Composition du Jury :

Jeroen BELIEN PhD, Professeur, HDR, KU Leuven	Rapporteur
Riad AGGOUNE PhD, Professeur associé, HDR, Luxembourg Institute of Science and Technology	Rapporteur
Emmanuel HAVEN BA, MA, PhD, Professeur, Memorial University of Newfoundland	Examinateur
Kristiaan KERSTENS PhD, Directeur de recherche, HDR, CNRS-IESEG	Président du jury
Christine DI MARTINELLY PhD, Professeur associé,HDR, IESEG	Examinateur
Stefan CREEMERS PhD, Professeur., HDR, IESEG	Directeur de thèse
Samuel DELEPLANQUE PhD, Professeur associé, JUNIA-ISEN	Invité
Tanja MLINAR PhD, Professeur associé, IESEG	Invité

Acknowledgements

I would like to express my deepest gratitude to my wife, Stephania, for her unwavering support during my darkest moments. Without her encouragement and belief in me, this journey would have been infinitely more difficult and most likely not possible.

I am profoundly grateful to my parents, my brother, and my sister. My family is my rock, providing me with the strength and motivation to persevere.

I want to specially express my gratitude towards my PhD supervisor, Professor Stefan Creemers, for his support and guidance over these years. His mentorship has helped me to grow as a student, researcher, instructor, and human being.

I am thankful to IESEG, LEM, the Ecole Doctorale SESAM, and the University of Lille for providing me with the opportunity to conduct my research and complete my PhD. Specially I am grateful to the faculty and research administrative staff of IESEG.

Special thanks to companies like IBM and D-Wave for granting cloud access to their quantum machines. Without this privilege, this thesis would not have been possible.

To my friends, thank you for being there for me and for all your help. Your support has been invaluable.

I would also like to thank Sci-Hub for providing free access to research articles that would otherwise be inaccessible behind paywalls, as well as open free research repositories like arXiv.

A big thank you to my dog Vittorio for his emotional support over these years. His presence has been a constant source of comfort. It is almost impossible to remain angry when you are staring at the face of a pug.

Finally, I extend my gratitude to the members of the jury for their feedback and willingness to participate as members of the jury for my dissertation.

Abstract

Quantum computing, with its profound potential, has garnered significant attention; however, its integration into Operations Research (OR) remains nascent. This dissertation attempts to address this gap by introducing innovative algorithms rooted in quantum computing principles to tackle discrete optimization problems. The first part focus on the fundamentals of quantum computing and analyzes Grover's algorithm. Perhaps the most important quantum algorithm that is renowned for its quadratic speedup in unstructured searches. Leveraging Grover's algorithm, the study presents a procedure that effectively solves the binary knapsack problem, overcoming challenges often overlooked in the quantum computing literature.

In the second part, the thesis refines the performance of the algorithms developed in the first chapter, introducing two novel procedures a hybrid branch-and-bound algorithm and a random-ascent algorithm. These procedures prove capable of matching the performance of leading classical algorithms across 66000 instances of the binary knapsack problem. Notably, they extend their applicability to discrete optimization problems lacking clear structures, showcasing the potential of quantum-inspired approaches. The refined procedures exhibit efficiency, solving any discrete optimization problem with at most $\mathcal{O}(\eta\sqrt{2}nb)$ operations.

In the third part the exploration extends to scrutinizing existing quantum algorithms' limitations in solving discrete optimization problems. The study critically evaluates quantum counting algorithms, as well as Grover's algorithm variants such as nested quantum search, quantum walks, and the amplitude amplification algorithm, revealing instances where classical counterparts may dominate.

In the fourth parth of this dissertation we explore a promising quantum computing avenue for solving complex combinatorial optimization problems, called quantum annealing and we specif-

ically use it to solve the Resource-Constrained Project Scheduling Problem (RCPSP). Formulating RCPSP as both a Quadratic Unconstrained Binary Optimization problem (QUBO) and an Ising Hamiltonian model, the study employs the commercially available D-Wave Advantage 6.3 quantum annealer, applied to RCPSP instances from the Random generator RANGEN1. Comparative assessments against classical solvers and metaheuristic algorithms affirm quantum annealing's potential, particularly for smaller to medium-sized RCPSP instances. The diverse landscape of solutions it offers makes quantum annealing a compelling candidate for hybrid hybrid optimization algorithms, promising to enhance solution quality.

This comprehensive thesis contributes to both the theoretical understanding and practical applications of quantum computing in Operations Research, showcasing its potential to revolutionize problem-solving methodologies.

Chapter five of this thesis explores a novel python open-source library dedicated to project scheduling called cheche-pm, that incorporates and make available scheduling methods from the literature to solve the RCPSP and it's variants. This library focuses on bringing a comprehensive collection of scheduling methods to both practitioners and researchers. The use of this library was critical for the development of chapter four of this thesis.

Finally Chapter six explores the usage of a hybrid quantum and classical implementation of quantum annealing to solve an special case of the quadratic semi-assignment problem QSAP, in the context of strategic group works. demonstrating the synergy between quantum and classical optimization strategies.

Résumé

L'informatique quantique, avec son potentiel profond, a suscité une attention significative ; cependant, son intégration dans la Recherche Opérationnelle (RO) reste naissante. Cette thèse tente de combler cette lacune en introduisant des algorithmes innovants ancrés dans les principes de l'informatique quantique pour résoudre des problèmes d'optimisation discrète. La première partie se concentre sur les fondamentaux de l'informatique quantique et analyse l'algorithme de Grover. Peut-être l'algorithme quantique le plus important qui est renommé pour son accélération quadratique dans les recherches non structurées. En tirant parti de l'algorithme de Grover, l'étude présente une procédure qui résout efficacement le problème du sac à dos binaire, surmontant des défis souvent négligés dans la littérature sur l'informatique quantique.

Dans la deuxième partie, la thèse affine la performance des algorithmes développés dans le premier chapitre, introduisant deux procédures novatrices : un algorithme hybride de branchement et de bornage et un algorithme d'ascension aléatoire. Ces procédures se révèlent capables d'égaler la performance des algorithmes classiques leaders sur 66 000 instances du problème du sac à dos binaire. Notamment, elles étendent leur applicabilité à des problèmes d'optimisation discrète manquant de structures claires, démontrant le potentiel des approches inspirées par le quantique. Les procédures affinées exhibent une efficacité, résolvant tout problème d'optimisation discrète avec au plus $\mathcal{O}(\eta\sqrt{2nb})$ opérations.

Dans la troisième partie, l'exploration s'étend à l'examen des limitations des algorithmes quantiques existants dans la résolution de problèmes d'optimisation discrète. L'étude évalue de manière critique les algorithmes de comptage quantique, ainsi que les variantes de l'algorithme de Grover telles que la recherche quantique imbriquée, les marches quantiques et l'algorithme

d'amplification d'amplitude, révélant des instances où les contreparties classiques peuvent dominer.

Dans la quatrième partie de cette thèse, nous explorons une avenue prometteuse de l'informatique quantique pour résoudre des problèmes complexes d'optimisation combinatoire, appelée recuit quantique et nous l'utilisons spécifiquement pour résoudre le Problème d'Ordonnancement de Projets avec Contraintes de Ressources (RCPSP). Formulant le RCPSP comme un problème d'Optimisation Binaire Quadratique Non Contrainte (QUBO) et un modèle d'Hamiltonien d'Ising, l'étude utilise l'annealer quantique D-Wave Advantage 6.3 disponible commercialement, appliqué à des instances RCPSP du générateur aléatoire RANGEN1. Les évaluations comparatives contre des solveurs classiques et des algorithmes métaheuristiques affirment le potentiel du recuit quantique, particulièrement pour des instances RCPSP de petite à moyenne taille. Le paysage diversifié de solutions qu'il offre fait du recuit quantique un candidat convaincant pour des algorithmes d'optimisation hybrides, promettant d'améliorer la qualité des solutions.

Cette thèse complète contribue à la compréhension théorique et aux applications pratiques de l'informatique quantique dans la Recherche Opérationnelle, démontrant son potentiel à révolutionner les méthodologies de résolution de problèmes.

Le chapitre cinq de cette thèse explore une nouvelle bibliothèque open-source en python dédiée à l'ordonnancement de projets appelée cheche-pm, qui intègre et rend disponibles des méthodes d'ordonnancement de la littérature pour résoudre le RCPSP et ses variantes. Cette bibliothèque se concentre sur l'apport d'une collection complète de méthodes d'ordonnancement à la fois aux praticiens et aux chercheurs. L'utilisation de cette bibliothèque a été critique pour le développement du chapitre quatre de cette thèse.

Enfin, le chapitre six explore l'utilisation d'une mise en œuvre hybride quantique et classique du recuit quantique pour résoudre un cas spécial

Contents

1 Fundamentals of Quantum Discrete Optimization **1**

1.1 Chapter 1 introduction 1

1.2 Discrete Optimization a Quantum Revolution (Part I) 3

1.2.1 Introduction 3

1.2.2 Fundamentals of Quantum Computing 6

1.2.3 Universal Quantum Computing 7

1.2.4 The Deutsch algorithm 10

1.2.5 Grover's algorithm 12

1.2.6 Using Grover's algorithm to solve the binary knapsack problem 15

1.2.7 Inherent problems of Grover's algorithm 24

1.2.8 Performance of procedure BSP 26

1.2.9 Discussion and conclusion 29

2 Discrete Optimization a Quantum Revolution **35**

2.1 Chapter 2 introduction 35

2.2 Discrete Optimization a Quantum Revolution (Part II) 37

2.2.1 Introduction 37

2.2.2 Hybrid branch-and-bound procedure 41

2.2.3 Random-ascent procedure 45

2.2.4 Improving procedures HBB and RAP 50

2.2.5 Combining procedures HBB and RAP 54

2.2.6 Generalizing procedures HBB and RAP 56

2.2.7 Conclusion . . . 57
2.2.8 Appendix . . . 60

3 Limitations of existing Quantum Discrete Optimization algorithms 65
3.1 Chapter 3 introduction . . . 65
3.2 Discrete optimization: Limitations of existing quantum algorithms . . . 67
3.2.1 Introduction . . . 67
3.2.2 Quantum counting algorithms . . . 70
3.2.3 Nested quantum search . . . 77
3.2.4 Amplitude amplification . . . 81
3.2.5 Conclusion . . . 83
3.2.6 Appendix . . . 85

4 Quantum discrete optimization: Heuristic approaches for solving the RCPSP 100
4.1 Chapter 4 introduction . . . 100
4.2 Solving the Resource-Constrained Project Scheduling Problem (RCPSP) with Quantum Annealing . . . 102
4.2.1 Introduction . . . 103
4.2.2 Quantum annealing and D-Wave machines . . . 104
4.2.3 Resource-constrained project scheduling Problem . . . 107
4.2.4 Benchmarking metrics for quantum annealing . . . 112
4.2.5 Methods . . . 112
4.2.6 Results . . . 119
4.2.7 Discussion . . . 135
4.2.8 Appendix . . . 138

5 cheche-pm: A Python Library for Project Scheduling 155
5.1 Chapter 5 introduction . . . 155
5.2 cheche_pm: a dedicated python library for project scheduling and project management . . . 156
5.2.1 Introduction . . . 156

5.2.2 Design 157
5.2.3 Example 159
5.2.4 Conclusions 165

6 Quantum discrete optimization: Hybrid quantum classical approach for strategic group formation 167

6.1 Chapter 6 introduction 167
6.2 Restricted Min-Norm Semi-Assignment in Group Formation: Quantum and Binary Algorithmic Integration 169
6.2.1 Introduction 170
6.2.2 Related literature 171
6.2.3 Model 173
6.2.4 Solution strategies 175
6.2.5 Quantum annealing for R-MN-SAP 179
6.2.6 Illustrative example 181
6.2.7 Empirical and computational analysis 184
6.2.8 Conclusion 191
6.2.9 Appendix 196

Chapter 1

Fundamentals of Quantum Discrete Optimization

The content of this chapter is presented in the form of a paper titled "**Discrete Optimization a Quantum Revolution (Part I)**"

1.1 Chapter 1 introduction

This chapter shows a systematic investigation into the complexities of quantum computing. Initially, it establishes the necessary theoretical background by elaborating on core principles fundamental to quantum computing, incorporating an exposition of essential concepts and the formal introduction of Dirac notation which is essential for further discussions. This foundational framework paves the way for an in-depth analysis of quantum gates and circuits. A necessary ingredient for achieving universal quantum computing capabilities.

Following this introduction, the chapter progresses to detail the Deutsch algorithm and its subsequent enhancement, the Deutsch-Josza algorithm. These algorithmic advancements represent significant achievements in the field of quantum computing, setting the stage for a detailed examination of Grover's algorithm, which is a primary focus of this chapter. A methodical breakdown follows, employing Grover's algorithm as an efficient mechanism for solving the

binary knapsack problem.

A considerable portion of this section is allocated to the exploration of challenges inherent to Grover's algorithm. This analysis thoroughly assesses the impact of these issues on the methodological efficacy. A detailed computational experiment provides critical insights into the operational runtime of this methodology when applied to 66000 instances of the binary knapsack problem.

The section concludes with a comparative evaluation, assessing the performance of this quantum computing-based methodology against traditional algorithms. This comparison offers a holistic view, leading to conclusions that highlight the key contributions and implications of this research within the wider context of quantum computing approaches to discrete optimization challenges.

Discrete optimization: A quantum revolution (Part I)

Stefan Creemers*
IESEG School of Management, Lille, France, s.creemers@ieseg.fr
KU Leuven, Leuven, Belgium, stefan.creemers@kuleuven.be

Pérez Luis Fernando
IESEG School of Management, Lille, France, l.perezarmas@ieseg.fr

Quantum computing has sparked a tremendous interest from governments, academics, and the private sector alike. In spite of this, quantum computing has received only limited attention in the Operations Research (OR) community. This is somewhat surprising given the potential that has been ascribed to quantum computers to solve complex OR problems. To investigate the potential of quantum computing from an OR perspective, we develop several new algorithms, and use them to solve discrete optimization problems. This paper is divided into two parts. In the first part, we introduce the fundamentals of quantum computing and provide an in-depth analysis of Grover's algorithm. Arguably, Grover's algorithm is one of the most important quantum algorithm as it achieves a quadratic speedup when performing an unstructured search. As a search algorithm, it can be used to find good/optimal solutions in a set of feasible solutions. We use Grover's algorithm as a subroutine in a procedure that is used to effectively solve the binary knapsack problem. The use of Grover's algorithm, however, is not without challenge. We identify a number of problems that are typically overlooked in the literature, and show how to overcome them. In addition, we define the conditions under which our procedure can outperform classical algorithms, and show that our procedure can almost match the performance of the best classical algorithms when solving the binary knapsack problem.

Keywords: Quantum; computing; algorithm; knapsack.

1. Introduction

The idea of quantum computing was first launched in 1980, when Benioff defines a quantum mechanical model of a Turing machine. A few years later, Feynman (1982) states the idea of a universal quantum simulator. Building on the work of Benioff and Feynman, Deutsch (1985) describes the first universal quantum computer that is able to efficiently simulate any other quantum computer. In addition, Deutsch also formulates the Deutsch algorithm, the first quantum algorithm that has a proven speedup when compared to a classical algorithm. Later, Deutsch and Josza (1992) extend this algorithm. Deutsch and Josza, however, consider a problem (the Deutsch-Josza problem) that has no practical use, and it takes until 1994 for quantum computing to really take off. In

*Corresponding author

1994, Shor presents a quantum algorithm to find the prime factors of large integers in polynomial time (whereas the best classical algorithm requires sub-exponential time). In theory, Shor's algorithm can be used to break many of the cryptography schemes in use today. Not surprisingly, the publication of Shor's algorithm sparked an enormous interest in quantum computing. A few year later, Grover (1996) published a quantum algorithm that performs an unstructured search, and that achieves a quadratic speedup when compared to a classical algorithm. Arguably, Grover's algorithm is the most important algorithm in quantum computing today. It can be used to solve a multitude of problems, and is the corner stone of many other quantum algorithms (see e.g., the Quantum Algorithm Zoo[1] that currently lists 64 quantum algorithms, many of which rely on Grover's algorithm). In what follows, we also focus on Grover-based quantum algorithms, and use them to solve discrete optimization problems.

We consider discrete optimization problems that assign discrete values to a set of n decision variables in order to optimize some objective value V. These problems can be expressed as follows:

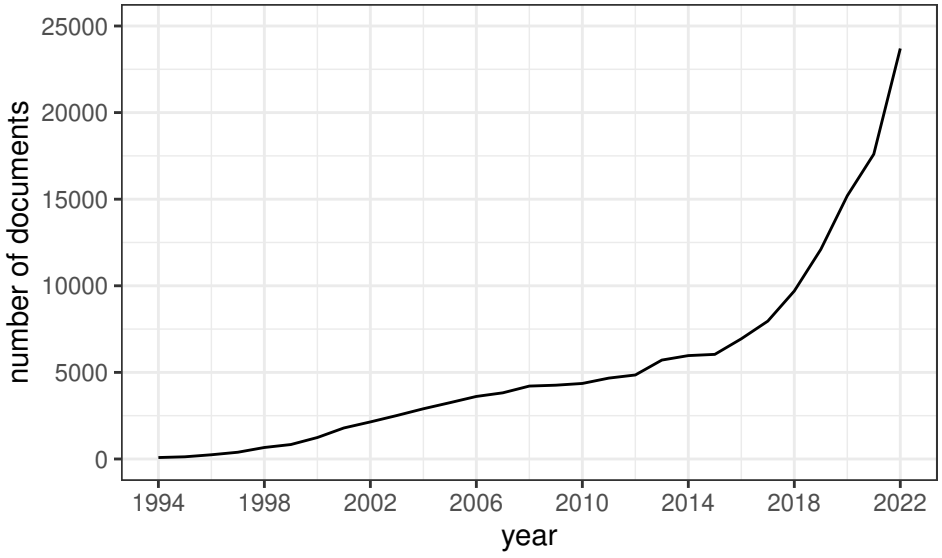
$$\begin{aligned} &\text{Maximize (or minimize)} \quad V \\ &\text{s.t.} \quad \mathbf{x} \in \mathcal{X} \end{aligned}$$

where $\mathbf{x} = \{x_1, \ldots, x_n\}$ is a solution (with x_i the discrete value that has been assigned to decision variable $i : 1 \leq i \leq n$), and $\mathcal{X}$ is the set of all feasible solutions that respect the logical constraints that define the optimization problem. Whereas it may be easy to evaluate the value and feasibility of a solution $\mathbf{x}$, it can be hard to determine the optimal solution $\mathbf{x}^*$ and its solution value V^*. Discrete optimization problems often have an underlying combinatorial structure, hence the link with combinatorial optimization (Parker and Rardin, 1988). Next to combinatorial optimization problems, integer programming problems are also a subset of discrete optimization problems, and can be solved using e.g., commercial solvers such as CPLEX and Gurobi (Wolsey, 2021).

Discrete optimization problems are at the core of OR, however, they are also mentioned as one of the four use cases of quantum computing by McKinsey (2021). According to McKinsey, the quadratic speedup offered by Grover's algorithm has potential applications in almost every industry. In addition, McKinsey (2023) estimates that quantum computing has an economic potential of up to $1,270 billion (to be realized by 2035 across four industries: chemicals, life sciences, finance, and automotive). They also show that $5.4 billion has already been invested in quantum computing, with companies such as IBM, Google, Amazon, and Microsoft that already offer commercial quantum-computing services. Next to the private sector, governments are also investing heavily in quantum technologies: McKinsey (2023) reports that governments have announced up to $34 billion of investments in quantum technologies. In addition, investments in quantum computing are expected to further increase in the upcoming years. This increasing trend can also be seen in the number of published papers that deal with quantum computing. For instance, Figure 1 shows the number of English documents that have been published since 1994 according to Google Scholar

[1] https://quantumalgorithmzoo.org/

Figure 1. Number of English documents that have been published according to Google Scholar when conducting a search for keyword "quantum computing".



when conducting a search using the keyword "quantum computing".

Notwithstanding the tremendous interest in quantum computing, the topic has only received limited attention from the OR community (with only a few publications in INFORMS journals that mention "quantum computing"). This is somewhat surprising given the potential that has been ascribed to quantum computers to solve complex optimization problems. In Part I of this two-part paper, we open up the field of quantum computing to the OR community, and investigate its potential to solve discrete optimization problems. For this purpose, we develop a new procedure, and use it to effectively solve the binary knapsack problem. The procedure uses Grover's algorithm as a subroutine, and is verified using Qiskit (IBM's software development kit that allows to simulate or operate a universal quantum gate computer). By effectively solving the binary knapsack problem, we are able to identify a number of problems that arise when using Grover's algorithm to solve discrete optimization problems. We demonstrate how to overcome these problems, and perform an extensive computational experiment that shows that our procedure can match the performance of the best classical algorithms when solving the binary knapsack problem if certain conditions are met. We also define the conditions under which our procedure can outperform classical algorithms in a more general setting. In Part II of this paper, we discuss more advanced quantum algorithms, and propose two new quantum procedures that establish the new state-of-the-art.

It is beyond the scope of this paper to provide an exhaustive overview of the literature on quantum computing (refer to the Quantum Algorithm Zoo, Nielsen and Chuang (2010), and Hidary (2021) for an overview of the most important results). In addition, even though heavily researched, we do not consider adiabatic quantum computation (see Albash and Lidar (2018) and Glover et al. (2022) for an introduction). Instead, we focus on Grover-based algorithms as they allow us to achieve our goal: to show that quantum computing can cause a revolution in the field of discrete

optimization. Note, however, that it may take many more years for this revolution to take place as it requires access to a fault-tolerant universal quantum computer that has sufficient scale. Even in the most ambitious roadmap, such a computer is not expected to hit the market before 2030 (refer to e.g., Preskill (2018), MacQuarrie et al. (2020), and Cheng et al. (2023) for a discussion).

In what follows, in Section 2, we introduce the fundamentals of quantum computing as well as the notation that will be used throughout the rest of the paper. Next, in Section 3, we define the gates and circuits that are required to establish universal quantum computing. In Section 4, we introduce the Deutsch algorithm and its extension to the Deutsch-Josza algorithm. The Deutsch-Josza algorithm serves as an important stepping stone towards Grover's algorithm, that is discussed in Section 5. In Section 6, we present a procedure that adopts Grover's algorithm as a subroutine, and use this procedure to effectively solve the binary knapsack problem. Section 7 identifies a number of problems inherent to Grover's algorithm. The impact of these problems on the performance of our procedure are discussed in Section 8. Section 8 also presents an extensive computational experiment that allows to assess the practical runtime of our procedure when solving the binary knapsack problem. After comparing the performance of our procedure with that of classical algorithms, Section 9 concludes.

2. Fundamentals of Quantum Computing

In 1939, Dirac introduced the "bra-ket" notation to facilitate the study of quantum systems. In this notation, a "ket" $|c\rangle$ denotes a column vector c defined in a complex vector space $\mathcal{V}$, a "bra" $\langle r|$ denotes a row vector r that acts as a linear map on $\mathcal{V}$, and a "bra-ket" $\langle r|c\rangle$ represents the inner product of $\langle r|$ and $|c\rangle$. In what follows, we adopt bra-ket notation.

In quantum computing, information is stored in so-called "qubits". A qubit can be seen as a two-state quantum system that has basis states $|0\rangle = (1,0)^{\mathsf{T}}$ and $|1\rangle = (0,1)^{\mathsf{T}}$. These basis states are the quantum equivalents of states 0 and 1 of a classical bit. In contrast to a classical bit, however, a qubit may also be in a superposition of both basis states (i.e., it is in both states at the same time). The state of a quantum system is described by a so-called "wave function". In case of a single qubit, the wave function is a linear combination of basis states $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{1}$$

where α and β are complex numbers that are also known as the "probability amplitudes" of wave function $|\psi\rangle$. By observing (i.e., measuring) a qubit in superposition, the state of the quantum system collapses into one of the basis states. The probability that a qubit collapses into basis state $|0\rangle$ or $|1\rangle$ is proportional to the square of probability amplitutes α and β, respectively. This result is also known as the Born rule (1926). Given that α and β are complex numbers, we have:

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2}$$

In systems with two (or more) qubits, the basis states can be obtained as the Kronecker product of the basis states of each individual qubit. For instance, if we consider two qubits, a first basis state is given by:

$$|00\rangle = |0\rangle \otimes |0\rangle = (1,0)^{\mathsf{T}} \otimes (1,0)^{\mathsf{T}} = (1,0,0,0)^{\mathsf{T}}.$$

The other basis states are $|01\rangle = (0,1,0,0)^{\mathsf{T}}$, $|10\rangle = (0,0,1,0)^{\mathsf{T}}$, and $|11\rangle = (0,0,0,1)^{\mathsf{T}}$. If both qubits are in superposition, we get:

$$|\mathbf{\Psi}\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = (\alpha_1 |0\rangle + \beta_1 |1\rangle) \otimes (\alpha_2 |0\rangle + \beta_2 |1\rangle) = \alpha_1\alpha_2 |00\rangle + \alpha_1\beta_2 |01\rangle + \beta_1\alpha_2 |10\rangle + \beta_1\beta_2 |11\rangle ,$$

where $|\mathbf{\Psi}\rangle$ is used to denote the state of a set of qubits.

3. Universal Quantum Computing

Similar to classical computing, quantum computing uses logic gates to perform operations. In contrast to the gates of classical computing, however, quantum gates are reversible, and are represented by unitary matrices (i.e., a matrix U is unitary if $UU^{-1} = UU^{\mathsf{T}} = I$, where I is the identity matrix). In what follows, we first introduce the X, CX, and Toffoli quantum gates, and show how these gates can be used to create the quantum equivalent of the AND, OR, XOR, and NOT gates of classical computing (note, however, that there are many other quantum gates; refer to e.g., Aaronson (2018) for an overview of other quantum gates).

The X gate is the quantum equivalent of the classical NOT gate, and transforms input state $|0\rangle$ into output state $|1\rangle$ (and vice versa). Formally, X is a two-by-two matrix that operates as follows:

$$\mathrm{X}|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle , \;\; \mathrm{X}|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle .$$

The CX (or controlled-X) gate has two input qubits: a target qubit on which an X operation may be performed, and a control qubit that determines whether or not the X operation is performed. If the control qubit is in basis state $|1\rangle$, the X operation is performed on the target qubit. Otherwise, if the control qubit is in basis state $|0\rangle$, the target qubit remains unchanged. Formally, CX is a four-by-four matrix that operates as follows:

$$\mathrm{CX}|00\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |00\rangle .$$

One can verify that $\mathrm{CX}|10\rangle = |11\rangle$, $\mathrm{CX}|01\rangle = |01\rangle$, and $\mathrm{CX}|11\rangle = |10\rangle$.

The Toffoli gate is a controlled-CX (CCX) gate that has two control qubits. Only if both control qubits are in basis state $|1\rangle$, the X operation is performed on the target qubit. Formally, CCX is

an eight-by-eight matrix that operates as follows:

$$\mathrm{CCX}\,|000\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |000\rangle\,.$$

One can verify that $\mathrm{CCX}\,|100\rangle = |100\rangle$, $\mathrm{CCX}\,|010\rangle = |010\rangle$, $\mathrm{CCX}\,|110\rangle = |111\rangle$, $\mathrm{CCX}\,|001\rangle = |001\rangle$, $\mathrm{CCX}\,|101\rangle = |101\rangle$, $\mathrm{CCX}\,|011\rangle = |011\rangle$, and $\mathrm{CCX}\,|111\rangle = |110\rangle$.

We can use the X, CX, and CCX gates to create circuits that mimic the classical AND, OR, XOR, and NOT gates: the quantum equivalent of the AND gate corresponds to a CCX gate with target qubit initialized as $|0\rangle$, the quantum equivalent of the OR gate is a series of a CX, X, CCX, and X gate with target qubit initialized as $|0\rangle$, the quantum equivalent of the XOR gate is a series of two CX gates with target qubit initialized as $|0\rangle$, and the quantum equivalent of the NOT gate is simply the X gate. The circuits of all aforementioned gates are shown in Figure 2.

The X, CX, and CCX gates suffice to perform any classical operation. Quantum computers, however, are not limited to classical operations. To establish universal quantum computing, we need to define a set of universal quantum gates; a set of gates to which any operation on a quantum computer can be reduced. For this purpose, we need to introduce yet another quantum gate: the Hadamard gate. Together with the CCX gate, the Hadamard gate forms a set of universal quantum gates, and allows to establish universal quantum computing (Aharonov, 2003). In addition, the Hadamard gate is one of the most important quantum gates that it is used to put qubits in superposition. In case of a single qubit, the Hadamard gate (represented by the letter "H") operates as follows:

$$\mathrm{H}\,|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle\,,\quad \mathrm{H}\,|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle\,.$$

Because H is an involutory matrix (i.e., a square matrix that is its own inverse), we have $\mathrm{H}\,|+\rangle = \mathrm{HH}\,|0\rangle = |0\rangle$ and $\mathrm{H}\,|-\rangle = \mathrm{HH}\,|1\rangle = |1\rangle$. Note that a qubit in state $|+\rangle$ or $|-\rangle$ has an equal probability to collapse into basis state $|0\rangle$ or $|1\rangle$ when measured. In general, when applied to a set of n qubits, the Hadamard gate is defined as the nth tensor power of H (denoted $\mathrm{H}^{\otimes n}$), and the probability to measure any of the 2^n basis states equals 2^{-n}. Often, the goal of a quantum algorithm is to increase the probability to measure the correct basis state by performing operations on qubits in superposition. If an operation is performed on a set of n qubits in superposition, the operation is essentially performed on all 2^n basis states at once (whereas a classical computer would require 2^n operations; one for each basis state). Note, however, that it is not possible to measure

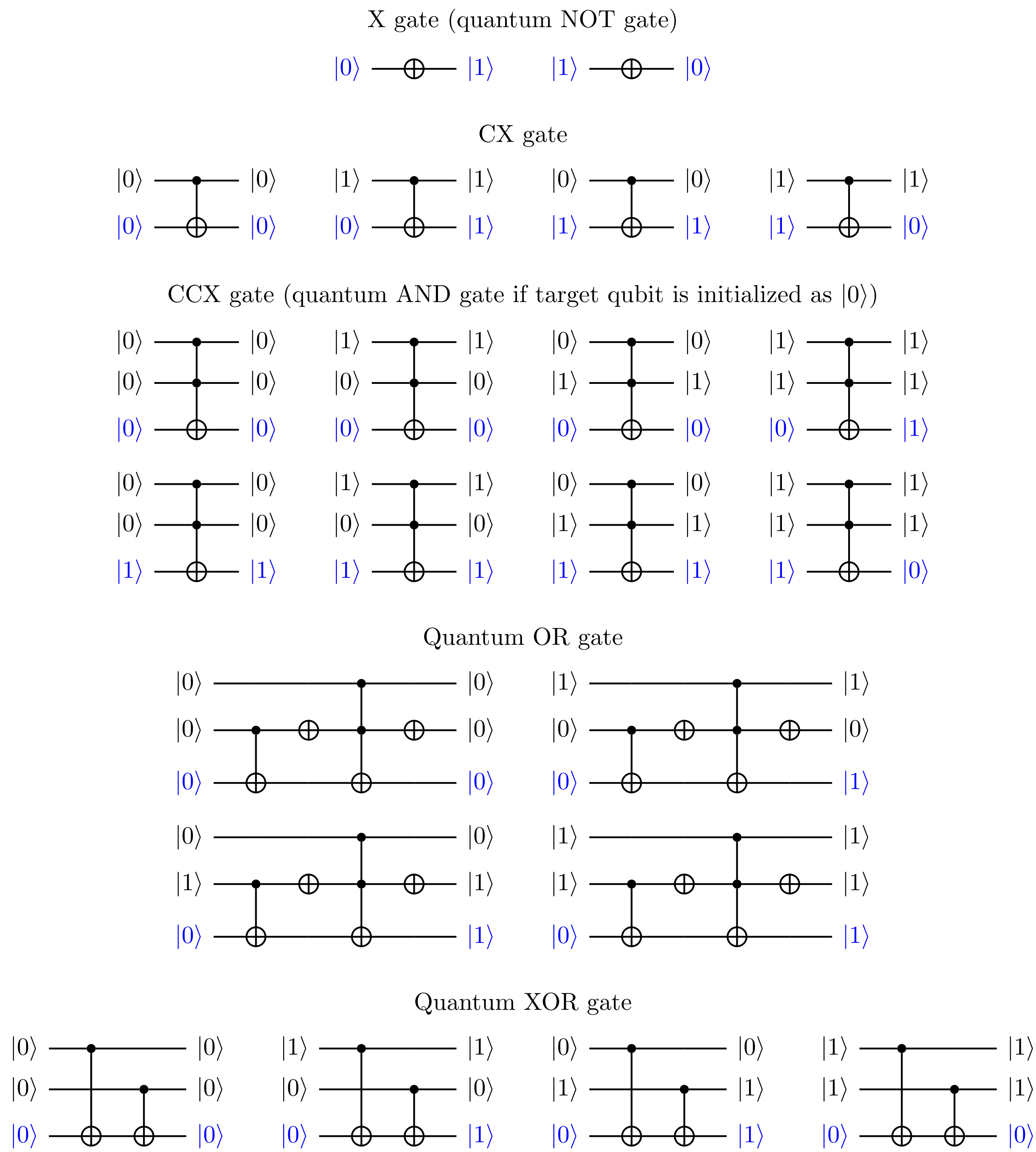


Figure 2. Circuits of the X, CX, and CCX quantum gates, as well as the quantum equivalents of the classical AND, OR, XOR, and NOT gates. The target qubit is indicated in blue.

only one (or a few) qubits in superposition; if a qubit in superposition is measured, the entire superposition collapses (which explains why quantum algorithms usually measure qubits after all operations have been performed; at the end of the procedure).

4. The Deutsch algorithm

The Deutsch algorithm was first published in 1985, has been generalized by Deutsch and Josza in 1992, and has further been improved by Cleve et al. (1998). Even though the Deutsch algorithm solves a problem that has no practical use, it is of particular interest to us because (1) it is the first quantum algorithms for which it has been shown that it can solve a problem more efficiently than a classical algorithm, and (2) it is a stepping stone towards Grover's algorithm that is introduced in the next section.

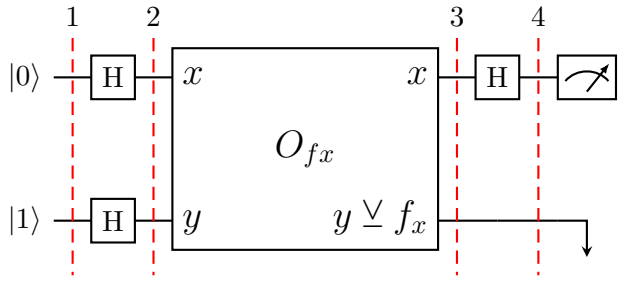
The Deutsch algorithm solves the following problem: we are given a function f_x that maps a binary input x to a binary output in a way that is unknown to us. Even though function f_x acts as a black box, we are told that f_x is either "constant" or "balanced"; f_x is said to be constant if it always produces the same output, and f_x is balanced if it produces a zero for half of the inputs and a 1 for the other half of the inputs. As such, f_x has to be one of the following four functions:

$$
\begin{aligned}
f_0 = 0 \text{ and } f_1 = 0 &\Rightarrow f \text{ is constant (case a)},\\
f_0 = 1 \text{ and } f_1 = 1 &\Rightarrow f \text{ is constant (case b)},\\
f_0 = 0 \text{ and } f_1 = 1 &\Rightarrow f \text{ is balanced (case c)},\\
f_0 = 1 \text{ and } f_1 = 0 &\Rightarrow f \text{ is balanced (case d)},
\end{aligned}
$$

To solve the Deutsch problem, it suffices to determine whether f_x is constant or balanced. The traditional way to solve this problem would be to evaluate both f_0 and f_1, and to observe whether the outputs are the same (i.e., f_x is constant if $f_0 = f_1$, and f_x is balanced otherwise). This solution, however, requires two calls to function f_x. Using the Deutsch algorithm, we only require a single call to function f_x.

The circuit of the Deutsch algorithm is presented in Figure 3, and can be broken down in four steps (indicated by the red dashed lines in the circuit). In a first step, we initialize two qubits, and

Figure 3. Circuit of the Deutsch algorithm



obtain quantum state $|\Psi_1\rangle = |01\rangle$. Next, in a second step, both qubits are put in superposition using a Hadamard gate:

$$|\Psi_2\rangle = \mathrm{H}^{\otimes 2}\,|01\rangle = \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{bmatrix}\begin{pmatrix}0\\1\\0\\0\end{pmatrix} = \frac{1}{2}\begin{pmatrix}1\\-1\\1\\-1\end{pmatrix} = \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle) = |+-\rangle\,.$$

In step 3, we pass both qubits through an oracle O_{fx} that, unlike us, has complete knowledge of function f_x. Oracle O_{fx} uses its knowledge of f_x to map state $|xy\rangle$ to $|x(y \veebar f_x)\rangle$, where $\veebar$ represents the XOR operation, and where x and y represent the first and second qubit, respectively. Note that O_{fx} only makes a single call to function f_x. After applying oracle O_{fx}, we get:

$$|\Psi_3\rangle = \frac{1}{2}\left(|0\rangle \otimes (|(0 \veebar f_0)\rangle - |(1 \veebar f_0)\rangle) + |1\rangle \otimes (|(0 \veebar f_1)\rangle - |(1 \veebar f_1)\rangle)\right).$$

For $x \in \{0,1\}$, one can verify that $|x\rangle \otimes (|0 \veebar f_x\rangle - |1 \veebar f_x\rangle) = (-1)^{f_x}\,|x\rangle \otimes (|0\rangle - |1\rangle)$, and hence, the sign (or phase) of the second qubit is "kicked back" to the first qubit if $f_x = 1$. As a result, the sign of the first qubit "flips", and $|\Psi_3\rangle$ may also be written as:

$$|\Psi_3\rangle = \frac{1}{2}\Big((-1)^{f_0}\,|0\rangle \otimes (|0\rangle - |1\rangle) + (-1)^{f_1}\,|1\rangle \otimes (|0\rangle - |1\rangle)\Big).$$

Quantum state $|\Psi_3\rangle$ can further be simplified depending on whether f_x is constant or balanced. For each of the four aforementioned cases, Table 1 lists the mappings of oracle O_{fx}.

Table 1. Mapping of basis states by oracle O_{fx} for each of the four cases.

case	$O_{fx}\,(\|00\rangle)$	$O_{fx}\,(\|01\rangle)$	$O_{fx}\,(\|10\rangle)$	$O_{fx}\,(\|11\rangle)$
a	$\|00\rangle$	$\|01\rangle$	$\|10\rangle$	$\|11\rangle$
b	$\|01\rangle$	$\|00\rangle$	$\|11\rangle$	$\|10\rangle$
c	$\|00\rangle$	$\|01\rangle$	$\|11\rangle$	$\|10\rangle$
d	$\|01\rangle$	$\|00\rangle$	$\|10\rangle$	$\|11\rangle$

Using these mappings, we get:

$$|\Psi_{3.a}\rangle = \frac{1}{2}\left(|00\rangle - |01\rangle + |10\rangle - |11\rangle\right) = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |+-\rangle\,,$$

$$|\Psi_{3.b}\rangle = \frac{1}{2}\left(-|00\rangle + |01\rangle - |10\rangle + |11\rangle\right) = -\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} = -|+-\rangle\,,$$

$$|\Psi_{3.c}\rangle = \frac{1}{2}\left(|00\rangle - |01\rangle - |10\rangle + |11\rangle\right) = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |--\rangle\,,$$

$$|\Psi_{3.d}\rangle = \frac{1}{2}\left(-|00\rangle + |01\rangle + |10\rangle - |11\rangle\right) = -\frac{|0\rangle - |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} = -|--\rangle\,.$$

In case a, the state of the system remains unchanged (i.e., $|\Psi_2\rangle = |\Psi_{3.a}\rangle$). In cases b and d, the global phase of the system changes (indicated in blue). The global phase of a system, however, has no impact on the state of the system; $|\Psi\rangle$ and $-|\Psi\rangle$ are said to be equivalent. In cases c and d, the local phase of the first qubit changes (indicated in red). In contrast to a change in global phase, a change in local phase may have an impact, as will become clear in the last step of the Deutsch algorithm. In the last step, we dispose of the second qubit (it is no longer needed), and observe that the first qubit is in state $|+\rangle$ if f_x is constant (cases a and b), and in state $|-\rangle$ if f_x is balanced (cases c and d). Therefore, if the first qubit is passed through a second Hadamard in the last step, we measure $|\Psi_4\rangle = \mathrm{H}|+\rangle = |0\rangle$ if f_x is constant, and $|\Psi_4\rangle = \mathrm{H}|-\rangle = |1\rangle$ if f_x is balanced. As such, the Deutsch algorithm is able to determine whether f_x is constant or balanced with only a single call to function f_x. This result was further generalized by Deutsch and Josza, who consider a function $f_{\mathbf{x}}$ that has n binary inputs $\mathbf{x} = \{x_1, \ldots, x_n\}$ (rather than only a single binary input, as was the case in the Deutsch algorithm).

5. Grover's algorithm

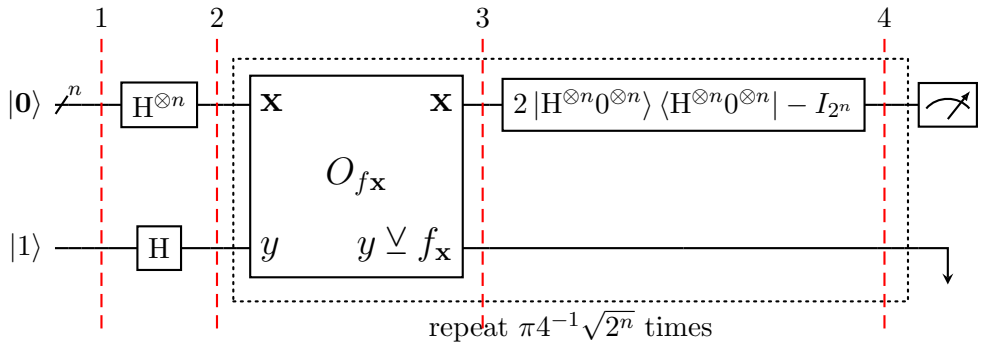
The goal of Grover's algorithm is to find a single target entry in an unstructured database that has 2^n entries. Similar to Deutsch and Josza, Grover considers a function $f_{\mathbf{x}}$ that has n binary inputs that are modeled using n qubits. The resulting quantum system has 2^n basis states that each correspond to an entry of the database. In contrast to Deutsch and Josza, however, $f_{\mathbf{x}}$ is not constant or balanced, but returns 1 if the target entry is given as an input, and 0 otherwise. Table 2 provides an example with three binary inputs, resulting in a quantum system that has eight basis states. In this example, function $f_{\mathbf{x}}$ outputs 1 only for target entry 101 (represented by basis state $|101\rangle$). Throughout the remainder of this section, this example will be used to illustrate the different steps of Grover's algorithm.

Table 2. Example database with eight entries that are represented by the basis states of a quantum system that has three qubits. The target entry is printed in blue. Function $f_{\mathbf{x}}$ returns 1 only for the target entry.

$\mathbf{x}$	000	100	010	110	001	101	011	111
$f_{\mathbf{x}}$	0	0	0	0	0	1	0	0

The circuit of Grover's algorithm is presented in Figure 4. Once again, the algorithm can be broken down in four steps (indicated by the red dashed lines in the circuit). The first steps of Grover's algorithm are almost identical to those of the Deutsch-Josza algorithm. The only difference is that oracle $O_{f\mathbf{x}}$ only flips the phase of the basis state that corresponds to the target entry (i.e., oracle $O_{f\mathbf{x}}$ flips the phase of the basis states for which $f_{\mathbf{x}}$ returns 1). In our example, $|\Psi_1\rangle = |0001\rangle$, $|\Psi_2\rangle = |+++-\rangle$, and $|\Psi_3\rangle$ is given by (the basis state whose phase has been flipped is indicated

Figure 4. Circuit of Grover's algorithm



in blue):

$$|\mathbf{\Psi}_3\rangle = \frac{1}{\sqrt{8}}\left(|000\rangle + |100\rangle + |010\rangle + |110\rangle + |001\rangle - |101\rangle + |011\rangle + |111\rangle\right) \otimes |-\rangle\,.$$

At this stage, if we observe the first three qubits, there is an equal probability that the system collapses into either one of the 8 basis states. In other words, we have 1 chance out of 8 to find out that $|101\rangle$ is the target basis state; measuring the state of the system at this stage is equivalent to randomly guessing the target entry. To increase the probability of measuring the correct basis state, Grover uses a so-called "difusion" operator. In short, the diffusion operator is a quantum gate that reflects the probability amplitudes of the basis states about the average probability amplitude. The goal of the diffusion operator is to amplify the probability amplitude of the target basis state. For three qubits, the diffusion operator is given by:

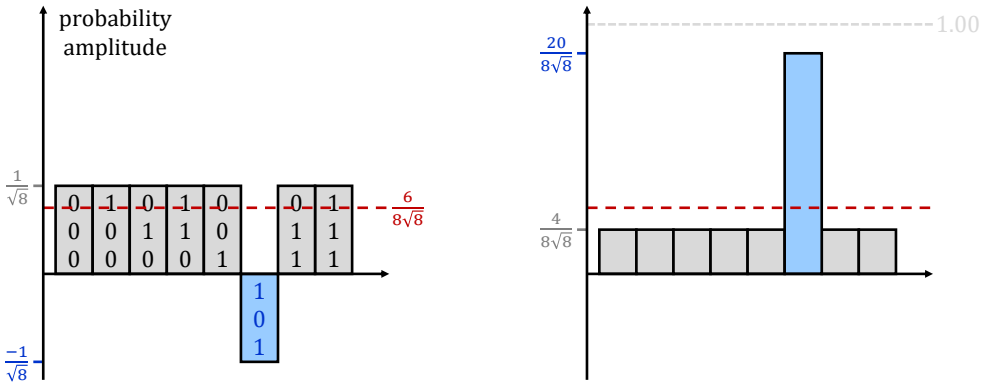
$$2\left|\mathrm{H}^{\otimes 3}0^{\otimes 3}\right\rangle\left\langle\mathrm{H}^{\otimes 3}0^{\otimes 3}\right| - I_8 = \begin{pmatrix} -0.75 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & -0.75 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & -0.75 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & -0.75 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 & -0.75 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & -0.75 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & -0.75 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & -0.75 \end{pmatrix},$$

where $|0^{\otimes n}\rangle$ is the nth tensor power of $|0\rangle$, $|c\rangle\langle r|$ is the outer product of $|c\rangle$ and $\langle r|$, and I_{2^n} is an identity matrix of dimension $2^n \times 2^n$. When applied to our example, one can verify that the average probability amplitude is $6(8\sqrt{8})^{-1}$, and we get:

$$|\mathbf{\Psi}_4\rangle = \left(\frac{20}{8\sqrt{8}}|101\rangle + \frac{4}{8\sqrt{8}}\left(|000\rangle + |100\rangle + |010\rangle + |110\rangle + |001\rangle + |011\rangle + |111\rangle\right)\right) \otimes |-\rangle\,.$$

In other words, the probability amplitude of the target basis state (i.e., $-(8\sqrt{8})^{-1}$) is reflected

Figure 5. Probability amplitudes of the different basis states before and after applying the diffusion operator in a first iteration (target basis state is colored blue, other basis states are colored gray, and the average amplitude used by the diffusion operator is colored red).



about $6(8\sqrt{8})^{-1}$, resulting in probability amplitude $20(8\sqrt{8})^{-1}$. The probability amplitudes of all other basis states (i.e., $(8\sqrt{8})^{-1}$) are also reflected, resulting in probability amplitudes $4(8\sqrt{8})^{-1}$. These reflections are illustrated in Figure 5.

After a first iteration, the probability of identifying the target basis state has increased from 0.125 to 0.7813 (i.e., $(20(8\sqrt{8})^{-1})^2$). We can further increase this probability by repeating steps 3 and 4. We get:

$$|\Psi_{3.2}\rangle = \left(-\frac{20}{8\sqrt{8}}|101\rangle + \frac{4}{8\sqrt{8}}\left(|000\rangle + |100\rangle + |010\rangle + |110\rangle + |001\rangle + |011\rangle + |111\rangle\right)\right) \otimes |-\rangle\,,$$

resulting in an average probability amplitude of $(8\sqrt{8})^{-1}$, and:

$$|\Psi_{4.2}\rangle = \left(\frac{22}{8\sqrt{8}}|101\rangle - \frac{2}{8\sqrt{8}}\left(|000\rangle + |100\rangle + |010\rangle + |110\rangle + |001\rangle + |011\rangle + |111\rangle\right)\right) \otimes |-\rangle\,.$$

After a second iteration, the probability of success increases to 0.9453 (i.e., $(20/8\sqrt{8})^2$); we are 94.53 percent certain to identify the target basis state after only two calls to function $f_{\mathbf{x}}$. Another iteration, however, will not increase the success probability:

$$|\Psi_{3.3}\rangle = \left(-\frac{22}{8\sqrt{8}}|101\rangle - \frac{2}{8\sqrt{8}}\left(|000\rangle + |100\rangle + |010\rangle + |110\rangle + |001\rangle + |011\rangle + |111\rangle\right)\right) \otimes |-\rangle\,,$$

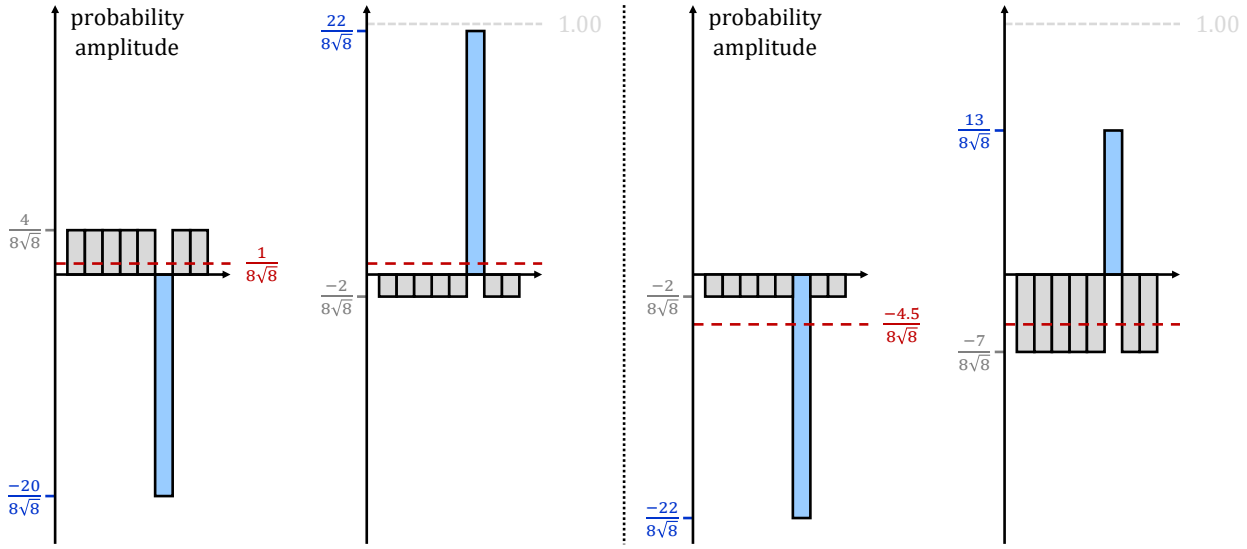
resulting in an average probability amplitude of $-4.5(8\sqrt{8})^{-1}$, and:

$$|\Psi_{4.3}\rangle = \left(\frac{13}{8\sqrt{8}}|101\rangle - \frac{7}{8\sqrt{8}}\left(|000\rangle + |100\rangle + |010\rangle + |110\rangle + |001\rangle + |011\rangle + |111\rangle\right)\right) \otimes |-\rangle\,.$$

After a third iteration, the probability of success drops to 0.3301. Figure 6 illustrates the second and third iteration.

For different numbers of iterations, Table 3 lists the average probability amplitude, the proba-

Figure 6. Probability amplitudes of the different basis states before and after applying the diffusion operator in a second (left panel) and third (right panel) iteration (target basis state is colored blue, other basis states are colored gray, and the average amplitude used by the diffusion operator is colored red).



bility amplitudes of the target and other basis states, and the probability to successfully identify the target basis state before the diffusion operator is applied. In addition, Figure 7 shows the probability amplitude of the target basis state after each iteration (i.e., after applying the diffusion operator). From Figure 7, one can observe that the probability amplitude of the target basis state can be approximated by a sine function that has its first extremum after two iterations. In fact, to find a target entry in a database of 2^n entries, Grover has shown that the probability amplitude of the target basis state reaches a first extremum after $\pi 4^{-1}\sqrt{2^n}$ iterations. In other words, whereas a classical algorithm requires at most 2^n calls to function $f_{\mathbf{x}}$, Grover's algorithm only needs $\pi 4^{-1}\sqrt{2^n}$ calls, resulting in a quadratic speedup.

Note that, even if there is more than one target entry, Grover's algorithm can still be used to identify one of the target entries. In general, if there are m target entries in a database of 2^n entries, Grover's algorithm needs $\pi 4^{-1}\sqrt{m^{-1}2^n}$ iterations to reach a first extremum. In addition, if the number of target entries is unknown, we can try different values of m until a matching entry has been found. One approach is to first evaluate Grover's algorithm assuming $m = 2^n$, and to evaluate $m = {}^m/_2$ in subsequent runs until a matching entry is found. In what follows, we adopt this approach, and use m and $\widehat{m}$ to denote the real and the assumed number of target entries, respectively. Note, however, that other approaches have also been discussed in the literature (see e.g., Boyer et al. (1996)).

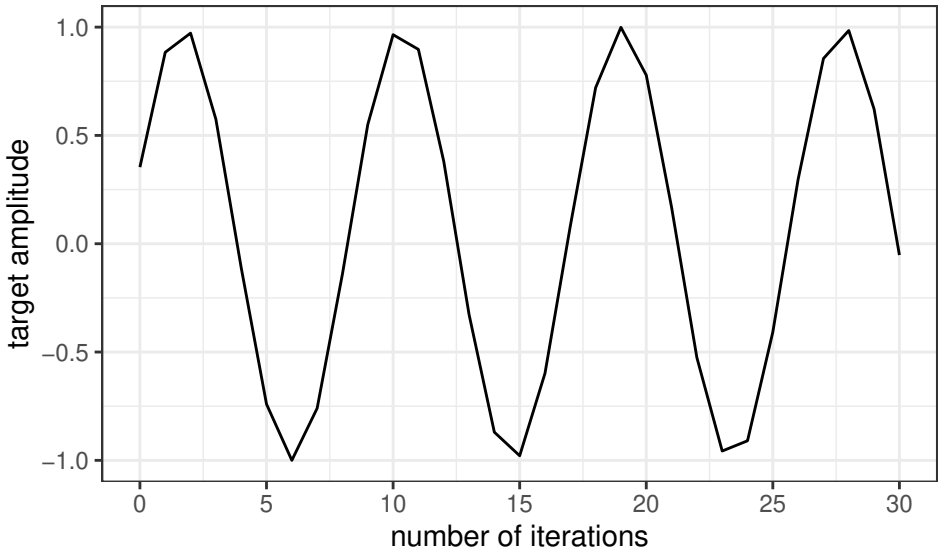
6. Using Grover's algorithm to solve the binary knapsack problem

In the previous section, we used a function $f_{\mathbf{x}}$ that outputs 1 if a set of n binary inputs corresponds to a given target entry in an unstructured database of 2^n entries. We can, however, also equip Grover's

Table 3. Average probability amplitude, probability amplitude of the target basis state, probability amplitude of the other basis states, and probability to successfully identify the target basis state before step 4 for different numbers of iterations.

iteration	average amplitude	target amplitude	other amplitude	success probability
1	0.2652	-0.3536	0.3536	0.1250
2	0.0442	-0.8839	0.1768	0.7813
3	-0.1989	-0.9723	-0.0884	0.9453
4	-0.3425	-0.5745	-0.3094	0.3301
5	-0.3149	0.1105	-0.3757	0.0122
6	-0.1298	0.7403	-0.2541	0.5480
7	0.1202	0.9999	-0.0055	0.9998
8	0.3100	0.7596	0.2458	0.5770
9	0.3449	0.1395	0.3743	0.0195
10	0.2073	-0.5504	0.3156	0.3029
11	-0.0339	-0.9650	0.0991	0.9313
12	-0.2582	-0.8972	-0.1669	0.8049
13	-0.3534	-0.3807	-0.3495	0.1450
14	-0.2719	0.3261	-0.3573	0.1063
15	-0.0544	0.8698	-0.1865	0.7566
16	0.1902	0.9787	0.0776	0.9578

Figure 7. Probability amplitude of the target basis state after each Grover iteration.



algorithm with other functions. By doing so, we can use Grover's algorithm to solve any discrete optimization problem. In this section, we use Grover's algorithm to solve the binary knapsack problem (we have chosen this problem because it is a typical discrete optimization problem that allows a clear illustration of the challenges that are faced when implementing Grover's algorithm; other discrete optimization problems can be solved just as well).

The binary knapsack problem is an NP-complete problem that has been studied for more than a century (see e.g., Kellerer et al. (2004)). The goal of the binary knapsack problem is to maximize the value of a knapsack by adding items from a pool of n items. Each item $i : 1 \leq i \leq n$ has a weight w_i and a value v_i. The weight of the selected items should not exceed W, the maximum weight capacity of the knapsack. The binary knapsack problem can be formulated as:

$$\text{Maximize} \quad \sum_{i=1}^{n} v_i x_i \tag{3}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} w_i x_i \leq W \tag{4}$$

$$x_i \in \{0, 1\}, \quad \forall\ 1 \leq i \leq n \tag{5}$$

In this formulation, x_i is a binary decision variable that determines whether or not item i has been added to the knapsack, objective function 3 maximizes the value of the knapsack, constraint 4 ensures that the items in the knapsack do not exceed the maximum weight, and constraint 5 ensures that decision variables are binary. In what follows, we assume $w_i, v_i \in \mathbb{Z}_{\geq 0}$ for all $i : 1 \leq i \leq n$.

To solve the binary knapsack problem, we equip Grover's algorithm with a function $f_{\mathbf{x}}$ that outputs 1 if knapsack $\mathbf{x} = \{x_1, \ldots, x_n\}$ is valid (and 0 otherwise). As a result, Grover's algorithm will only flip the phase of basis state $|\mathbf{x}\rangle$ if knapsack $\mathbf{x}$ is valid. Knapsack $\mathbf{x}$ is valid if it is feasible (i.e., if it satisfies constraint 4), and if it has a value of at least V. To ensure that knapsack $\mathbf{x}$ has a value of at least V, we impose an additional constraint:

$$\sum_{i=1}^{n} v_i x_i \geq V \tag{6}$$

Because the optimal knapsack and its value (denoted x^* and V^*, respectively) are unknown, we adopt a binary search procedure that evaluates different values V. An outline of the procedure is provided in Algorithm 1 (hereafter referred to as procedure `BSP`; note that, with a minimum of adaptations, this procedure can also be used to solve other discrete optimization problems). After initializing a minimum knapsack value $V_{\min}$ and a maximum knapsack value $V_{\max}$, procedure `BSP` evaluates $L = \lceil \log_2(1 + (V_{\max} - V_{\min})) \rceil$ iterations of an outer loop. In each iteration of the outer loop, we initialize a knapsack value V and the assumed number of valid solutions $\widehat{m}$. V is initialized as the largest integer that is smaller-than-or-equal-to the center of the interval that has endpoints $V_{\min}$ and $V_{\max}$. In addition, we let $\widehat{m} = 2^n$ because we don't know the number of valid solutions (as also explained at the end of the previous section). Next, we perform at most $(n + 1)$ iterations

of an inner loop. In each of these iterations, we perform $I(n, \widehat{m}) = \left\lceil \pi 4^{-1} \sqrt{\widehat{m}^{-1} 2^n} \right\rceil$ iterations of Grover's algorithm, and measure the basis state into which the system collapses (i.e., we measure $|\mathbf{x}\rangle$). Because the outcome of Grover's algorithm is probabilistic, it is possible that solution $\mathbf{x}$ is not valid (e.g., solution $\mathbf{x}$ may have a weight that exceeds W). In addition, if no valid solution exists, any basis state that we measure will correspond to an invalid solution. Therefore, we need to verify whether $\mathbf{x}$ is valid; we need to verify whether $\sum_{i=1}^{n} w_i x_i \leq W$ and $\sum_{i=1}^{n} v_i x_i \geq V$. If the validity of solution $\mathbf{x}$ has been verified, $V_{\min}$ is updated (i.e., $V_{\min} = V + 1$), we break the inner loop, and start a new iteration of the outer loop in which we reinitialize V and $\widehat{m}$. If, on the other hand, solution $\mathbf{x}$ is not valid, there are two options. If $\widehat{m} > 1$, we let $\widehat{m} = \widehat{m}/2$, and perform another iteration of the inner loop. If, however, we already processed all values of $\widehat{m}$, we update $V_{\max}$ (i.e., $V_{\max} = V - 1$), we break the inner loop, and start a new iteration of the outer loop in which we reinitialize V and $\widehat{m}$. This process continues until $V_{\min} > V_{\max}$. The last found solution and its value are optimal. Note that, in order to reduce the number of iterations, lower and upper bound procedures may be used to initialize $V_{\min}$ and $V_{\max}$.

Algorithm 1: Binary search procedure (`BSP`) used to solve the binary knapsack problem using Grover's algorithm.

```
initialize V_min = 0 and V_max = Σ_{i=1}^{n} v_i;
do
    V = ⌊0.5(V_min + V_max)⌋;
    m̂ = 2^n;
    do
        perform I(n, m̂) iterations of Grover's algorithm equipped with f_x;
        measure basis state |x⟩;
        if f_x = 1 then
            update best-found solution x* = x and its value V* = V;
            V_min = V + 1;
            break;
        else if m̂ > 1 then
            m̂ = m̂/2;
        else
            V_max = V − 1;
            break;
    while m̂ ≥ 1;
while V_min ≤ V_max;
```

In the literature on quantum computing it is common to assume the existence of an oracle function that has complete knowledge of all valid solutions. In practice, however, such an oracle needs to effectively implement $f_{\mathbf{x}}$ in order to verify which solutions are valid. The implementation of $f_{\mathbf{x}}$ as a quantum circuit is presented in Figure 8. The circuit may be divided into three parts (indicated by the red dashed lines in the circuit). In the first two parts, we determine whether knapsack $\mathbf{x}$ satisfies constraints 4 and 6, respectively. In a third part, we obtain the output of

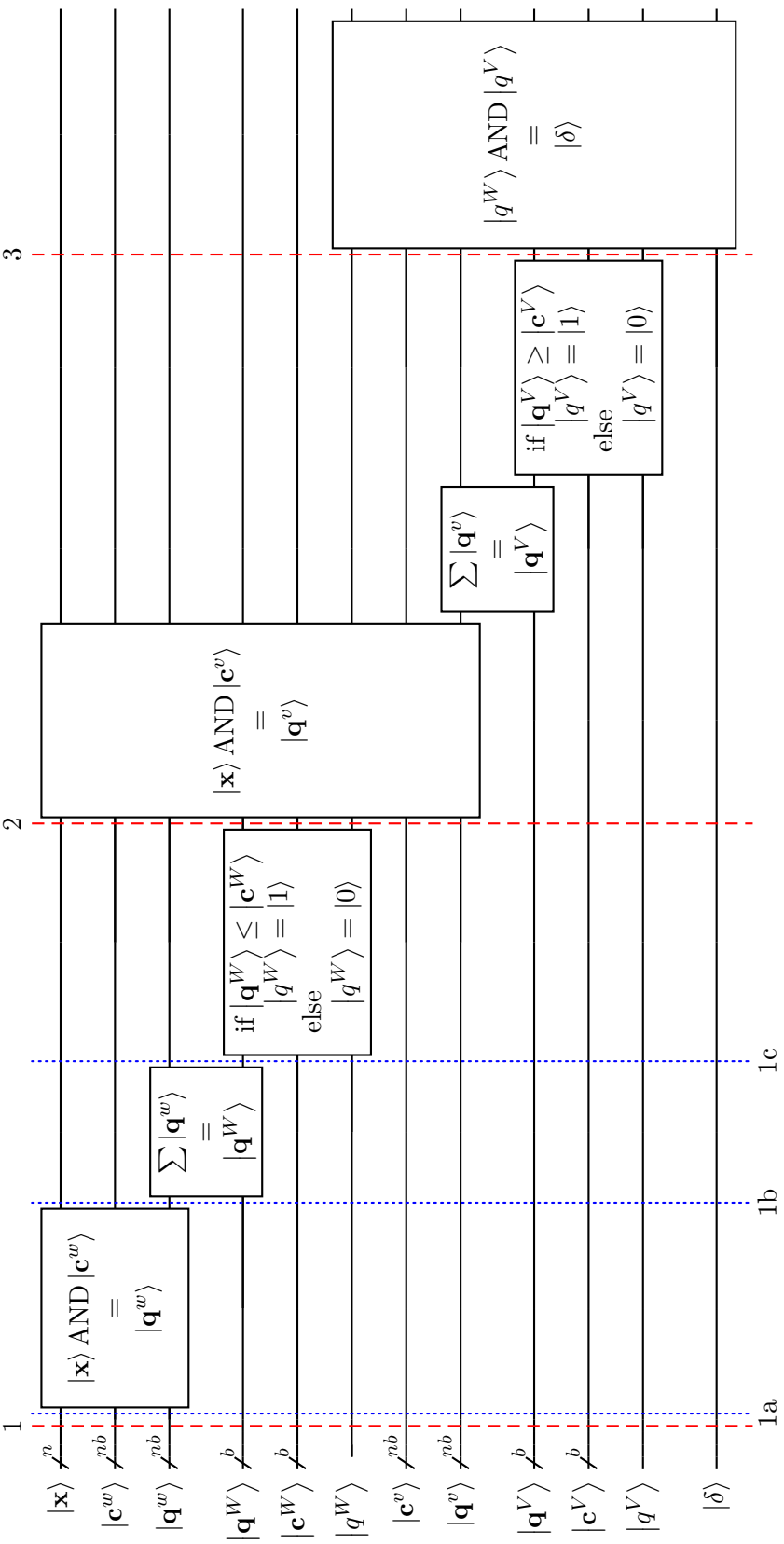


Figure 8. Circuit of function $f_{\mathbf{x}}$ that is used to verify whether a knapsack $\mathbf{x}$ has a weight smaller than W and a value of at least V.

function $f_{\mathbf{x}}$. In what follows, we use an example to demonstrate the dynamics of the circuit. In the example, we have a pool of three items and a knapsack that has maximum weight capacity $W = 4$. The weights and values of the items are listed in Table 4. Since we are working with qubits, all weights and values need to be converted to binary values. Let b denote the number of qubits that are required to represent the largest weight or value used in the example. In our example, $b = 3$, and the qubits that represent the maximum weight capacity are initialized as follows: $\left|c_1^W\right\rangle = |1\rangle$, $\left|c_2^W\right\rangle = |0\rangle$, and $\left|c_3^W\right\rangle = |0\rangle$ (with $\mathbf{c}^W = \left\{c_1^W, \ldots, c_b^W\right\}$). The binary conversions of the weights and values of the items are listed in Table 4, where $\mathbf{c}^w = \{\mathbf{c}_1^w, \ldots, \mathbf{c}_n^w\}$, $\mathbf{c}^v = \{\mathbf{c}_1^v, \ldots, \mathbf{c}_n^v\}$, $\mathbf{c}_i^w = \{c_{i1}^w, \ldots, c_{ib}^w\}$, and $\mathbf{c}_i^v = \{c_{i1}^v, \ldots, c_{ib}^v\}$. In our example, $\left|\mathbf{c}^W\right\rangle$, $|\mathbf{c}^w\rangle$, and $|\mathbf{c}^v\rangle$ are initialized as

Table 4. Data for example knapsack problem with three items.

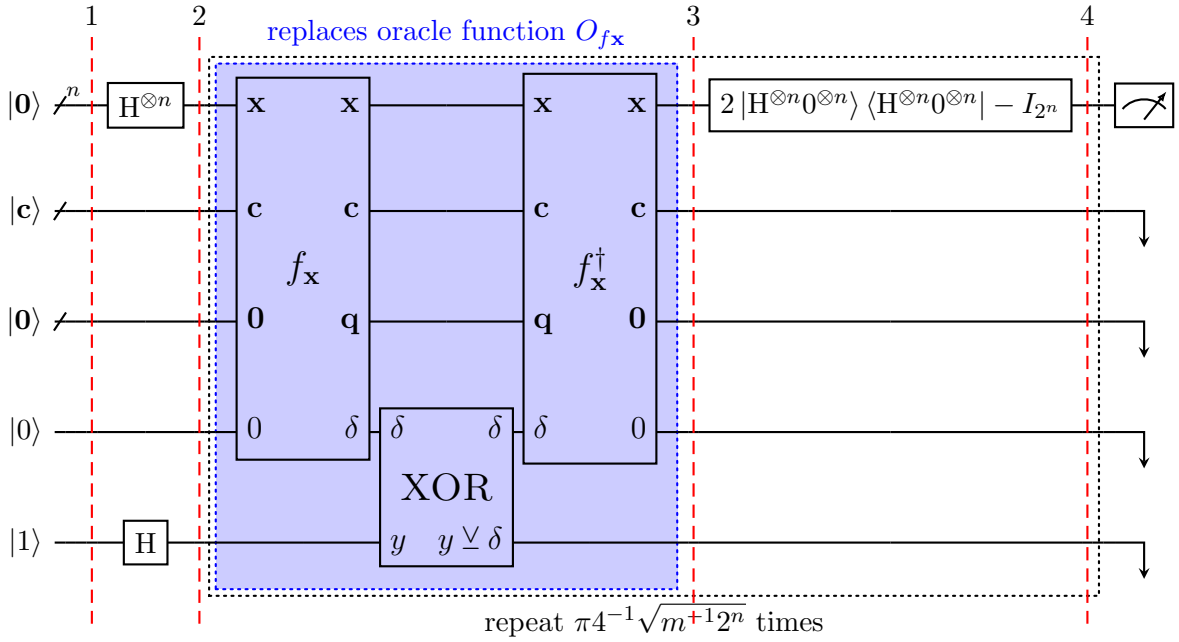
i	w_i	c_{i1}^w	c_{i2}^w	c_{i3}^w	v_i	c_{i1}^v	c_{i2}^v	c_{i3}^v
1	2	0	1	0	3	0	1	1
2	3	0	1	1	1	0	0	1
3	2	0	1	0	2	0	1	0

$|100\rangle$, $|010011010\rangle$, and $|011001010\rangle$, respectively. Next, imagine that we use the circuit of function $f_{\mathbf{x}}$ to evaluate knapsack $\mathbf{x} = \{1, 0, 1\}$. In this case, $|\mathbf{x}\rangle$ is initialized as $|101\rangle$. In a first part of the circuit, we distinguish three steps (indicated by the blue dotted lines in the circuit). In a first step (1a), we determine how much weight each item contributes to knapsack $\mathbf{x}$. The contributed weight of the items is represented by $\mathbf{q}^w = \{\mathbf{q}_1^w, \ldots, \mathbf{q}_n^w\}$, with $\mathbf{q}_i^w = \{q_{i1}^w, \ldots, q_{ib}^w\}$. $|\mathbf{q}^w\rangle$ is initialized as $|\mathbf{0}\rangle$, and is used to store the result of an AND operation on $|\mathbf{x}\rangle$ and $|\mathbf{c}^w\rangle$. In our example, one can verify that $|\mathbf{q}^w\rangle = |010000010\rangle$. In a second step (1b), we obtain the weight of knapsack $\mathbf{x}$ itself (represented by $\mathbf{q}^W = \left\{q_1^W, \ldots, q_b^W\right\}$). $\left|\mathbf{q}^W\right\rangle$ is initialized as $|\mathbf{0}\rangle$, and is used to store the result of $\sum_{i=1}^n |\mathbf{q}_i^w\rangle$. To perform the aforementioned addition, we can create our own adder circuits or use a more efficient circuit that was proposed by Draper (2002). In our example, $\left|\mathbf{q}^W\right\rangle = |100\rangle$. As a last step (1c), we verify whether knapsack $\mathbf{x}$ satisfies constraint 4. This can be done using the quantum equivalent of a digital magnitude comparator (see e.g., Xia et al (2018)). The digital magnitude comparator compares the weight of knapsack $\mathbf{x}$ (represented by $\mathbf{q}^W$) and the maximum knapsack weight (represented by $\mathbf{c}^W$), and outputs 1 if a smaller-than-or-equal-to condition is met. The result is stored in a qubit that is initialized as $\left|q^W\right\rangle = |0\rangle$. In our example, constraint 4 is satisfied, and $\left|q^W\right\rangle$ evaluates to $|1\rangle$. This concludes the first part of the circuit. The second part of the circuit is almost identical to the first part, and verifies whether the value of the knapsack (stored in $\left|\mathbf{q}^V\right\rangle = \sum_{i=1}^n |\mathbf{q}_i^v\rangle$, where $\mathbf{q}^V = \left\{q_1^V, \ldots, q_b^V\right\}$, $\mathbf{q}^v = \{\mathbf{q}_1^v, \ldots, \mathbf{q}_n^v\}$, and $\mathbf{q}_i^v = \{q_{i1}^v, \ldots, q_{ib}^v\}$) is at least equal to value V (stored in $\left|\mathbf{c}^V\right\rangle$, where $\mathbf{c}^V = \left\{c_1^V, \ldots, c_b^V\right\}$). If this is the case, constraint 6 is satisfied, and $\left|q^V\right\rangle = |1\rangle$. If, however, knapsack $\mathbf{x}$ does not satisfy constraint 6, then $\left|q^V\right\rangle = |0\rangle$. In a last part of the circuit, we obtain the outcome of function $f_{\mathbf{x}}$ itself (stored in a qubit that is initialized as $|\delta\rangle = |0\rangle$). If both constraints are satisfied, $|\delta\rangle = |1\rangle$ (and $|\delta\rangle = |0\rangle$ otherwise).

Figure 9 presents the circuit of Grover's algorithm equipped with function $f_{\mathbf{x}}$ (once more, the steps of the algorithm are indicated by red dashed lines). In this circuit, $\mathbf{x}$, $\mathbf{c}$, $\mathbf{q}$, and δ are the

qubits that are used to store the decision variables, the constant values that are used by function $f_{\mathbf{x}}$, the outcomes of operations performed by function $f_{\mathbf{x}}$, and the result of function $f_{\mathbf{x}}$, respectively. In addition, y is a qubit that is used to induce phase kickback. $|y\rangle$ is initialized as $|1\rangle$. The initialization of $|\mathbf{c}\rangle$ depends on the problem instance. All other qubits are initialized as $|0\rangle$. The gates grouped in blue replace oracle function $O_{f\mathbf{x}}$ in the original circuit of Grover's algorithm (see Figure 4). These gates perform three operations. First, function $f_{\mathbf{x}}$ evaluates whether $\mathbf{x}$ is valid, and stores the result in $|\delta\rangle$. Next, a XOR operation is used to map $|\delta y\rangle$ to $|\delta(y \veebar \delta)\rangle$. After that, function $f_{\mathbf{x}}^{\dagger}$ applies all gates of function $f_{\mathbf{x}}$ in reverse order. By doing so, function $f_{\mathbf{x}}^{\dagger}$ reinitializes $|\mathbf{q}\rangle$ and $|\delta\rangle$ (as $|\mathbf{0}\rangle$ and $|0\rangle$, respectively) such that they are ready for a next Grover iteration (note that $\mathbf{c}$ and y are unaffected, and remain in state $|\mathbf{c}\rangle$ and $|-\rangle$ until they are discarded at the end of the circuit). This process is ensures that the circuit is reversible, and is also known as "uncomputation".

Figure 9. Circuit of Grover's algorithm equipped with function $f_{\mathbf{x}}$.



We can now use procedure `BSP` to find the optimal solution to our example knapsack problem. The steps of `BSP` are illustrated in Figure 10. First, we let $V_{\min} = 0$ and $V_{\max} = \sum_{i=1}^{n} v_i = 6$. Next, we initiate a first iteration of the outer loop, and initialize $V = \lfloor 0.5(V_{\min} + V_{\max}) \rfloor = 3$ and $\widehat{m} = 2^n = 8$. In a first iteration of the inner loop, we evaluate $I(n, \widehat{m}) = 1$ Grover iteration. Given $V = 3$, there are two valid solutions (i.e., solution $\{1, 0, 0\}$ and $\{1, 0, 1\}$). As such, the phases of basis states $|100\rangle$ and $|101\rangle$ are flipped by Grover's algorithm. One can verify that, after a single Grover iteration, we have a 50 percent probability to measure $|100\rangle$, and a 50 percent probability to measure $|101\rangle$. In other words, if we observe the system after one Grover iteration, we are certain to measure a valid solution that has at least value $V = 3$. After verifying that $\{1, 0, 0\}$ (or $\{1, 0, 1\}$) is a valid solution, we let $V_{\min} = V + 1 = 4$, update the best-found solution, break the inner loop, and start a second iteration of the outer loop. In this second iteration, we let $V = 5$

and $\widehat{m} = 8$. Next, we initiate a first iteration of the inner loop, and again evaluate $I(n, \widehat{m}) = 1$ Grover iteration. Given $V = 5$, however, one can verify that there is only one valid solution (i.e., solution $\{1, 0, 1\}$). If there is only a single valid solution, a single Grover iteration only has a 78.13 percent chance to measure the target basis state (see also the previous section and Table 3). In other words, there is a 21.88 percent chance that we measure a basis state that is not valid. This illustrates why every solution resulting from Grover's algorithm needs to be verified. If the validity of the solution has been verified, we update the best-found solution, let $V_{\text{min}} = 6$, break the inner loop, and start a third iteration of the outer loop in which $V = 6$ (in this third iteration, no valid solution can be found for any value of $\widehat{m}$, and we let $V_{\text{max}} = V - 1 = 5$; as a result, $V_{\text{min}} > V_{\text{max}}$, and the procedure stops). If, on the other hand, no valid solution was identified (with probability 0.2188), we let $\widehat{m} = \widehat{m}/2 = 4$, and we perform $I(n, \widehat{m}) = 1$ Grover iteration. Again, we have a 78.13 percent chance to measure the basis state that corresponds to valid solution $\{1, 0, 1\}$. As such, we have a 17.09 percent chance to end up a situation where we did not measure a valid solution in the first iteration of the inner loop, but did measure a valid solution in the second iteration of the inner loop. If a valid solution was measured, we update the best-found solution, let $V_{\text{min}} = 6$, break the inner loop, and start a third iteration of the outer loop in which $V = 6$. If no valid solution was measured, we let $\widehat{m} = 2$, and perform $I(n, \widehat{m}) = 2$ Grover iterations. With 2 iterations, we have a 94.53 percent chance to identify a valid solution. If, however, we still fail to identify a valid solution, we let $\widehat{m} = 1$, and again perform $I(n, \widehat{m}) = 2$ Grover iterations. In total, we are 99.99 percent certain to measure $|101\rangle$, and to obtain $\{1, 0, 1\}$ as a valid solution during the second iteration of the outer loop. There is, however, a 0.0143 percent probability to measure the wrong basis state. If this happens, verification of the solution fails, we cannot reduce $\widehat{m}$ any further, and we let $V_{\text{max}} = V - 1 = 4$ (i.e., we have wrongfully concluded that $V = 5$ has no valid solution; this illustrates why it is possible that we are unable to identify the optimal solution if Grover's algorithm is used in an optimization procedure). At this point, we enter a third iteration of the outer loop with $V = 4$ (here again, we have a 99.99 percent probability to identify $\{1, 0, 1\}$ as the only valid solution that has value $V \geq 4$, after which we update V_{min} such that $V_{\text{min}} > V_{\text{max}}$, and the procedure stops; conversely, we have a 0.0143 percent probability to identify an invalid solution, after which we update V_{max} such that $V_{\text{min}} > V_{\text{max}}$, and the procedure also stops). Even though we are almost certain to find the optimal solution, we also require at least 8 Grover iterations (and at most 13). In addition, we need to verify the validity of at least 6 knapsack solutions (and at most 9). For each Grover iteration, we require two calls to function $f_{\mathbf{x}}$ (one call to function $f_{\mathbf{x}}$ and one call to function $f_{\mathbf{x}}^{\dagger}$ to uncompute). Each validation also requires a call to function $f_{\mathbf{x}}$. Therefore, to solve the example knapsack problem using `BSP`, we need at least 22 calls to function $f_{\mathbf{x}}$, and at most 35 (note that a brute-force classical approach would require us to evaluate only 8 knapsack solutions).

Figure 10. Illustration of the dynamics of procedure `BSP` for the example knapsack problem. In each node (represented by a square), we perform 1 to 2 Grover iterations (depending on the value of $\widehat{m}$), we measure the basis state, and we verify whether the corresponding solution is valid. If the solution is valid (indicated in green), $V_{\min}$ is updated. If, on the other hand, the solution is not valid (indicated in red), we update $\widehat{m}$ or $V_{\max}$ (if $\widehat{m} = 1$). The procedure stops if $V_{\min} > V_{\max}$ (indicated in blue).

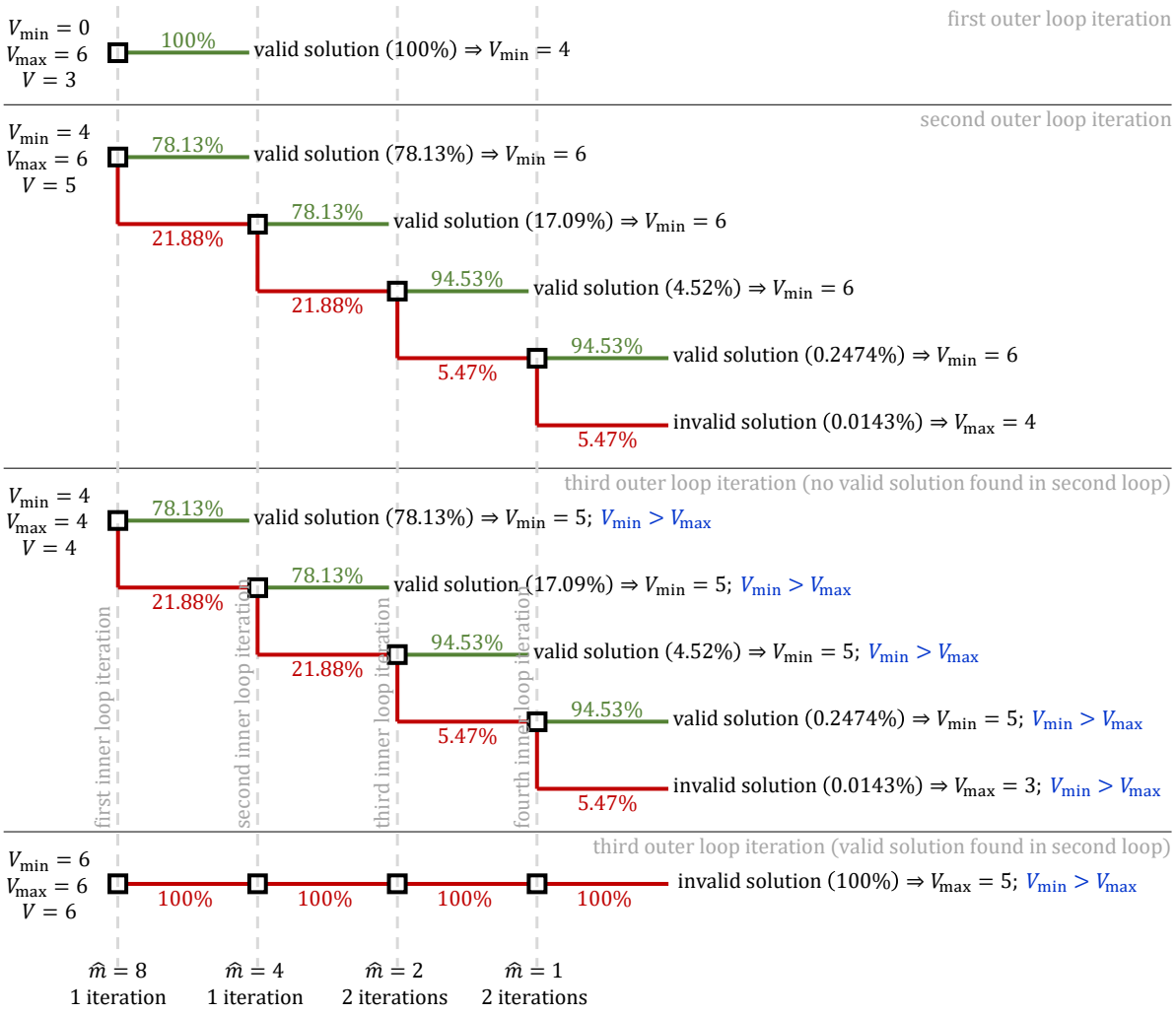
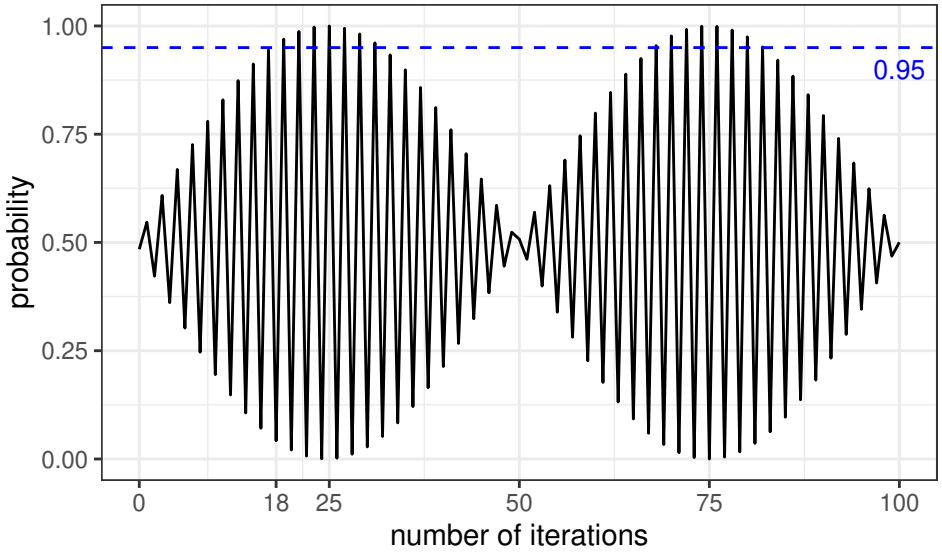


Figure 11. Probability of measuring a target basis state after each Grover iteration for a system that has 6 qubits and 31 target basis states (for reference, probability 0.95 is indicated by the blue dashed line).



7. Inherent problems of Grover's algorithm

In the previous Section, we have seen that the use of Grover's algorithm does not automatically lead to a quadratic speedup when solving discrete optimization problems. To make matters worse, Grover's algorithm has a number of other inherent problems. First, if we use Grover's algorithm to find one of the m target basis states in a database that has 2^n entries, we may need far more than $\pi 4^{-1}\sqrt{m^{-1}2^n}$ Grover iterations. For instance, if $m > {}^{2^n}/_2$, it is well known that approximation $\pi 4^{-1}\sqrt{m^{-1}2^n}$ may result in a low probability to measure a target basis state (e.g., in a system that has three qubits and six target basis states, $I(3,6) = 1$, and the probability to measure a target state after 1 Grover iteration equals 0). To resolve this problem, it has been suggested to add a "dummy" qubit. By adding a dummy qubit, the number of basis states effectively doubles while the number of target basis states remains the same. As a result, the number of target basis states is always smaller than (or equal to) half of the number of basis states. Doubling the number of basis states, however, also doubles the search space. As such, this is not an ideal (nor elegant) solution. In addition, even if $m \leq {}^{2^n}/_2$, approximation $\pi 4^{-1}\sqrt{m^{-1}2^n}$ may result in a low probability to measure a target basis state. For instance, consider a system that has six qubits that has 31 target basis states (and 33 other basis states). In such a system, we only have a 54.68 percent probability to measure a target basis state after $I(6,31) = 1$ Grover iteration. In order to be at least 95 percent certain to measure a target basis state, we need 18 Grover iterations. This is also illustrated in Figure 11 that shows the probability to measure one of the 31 target basis states after each Grover iteration. Unfortunately, the number of required Grover iterations is rather sensitive to the number of target basis states. For instance, in the example of 6 qubits, if we have 30, 29, 28, or 27 target basis states, we need 10, 7, 5, and 5 Grover iterations to be at least 95 percent

certain to measure any of the target basis states, respectively (refer to the Appendix for additional examples).

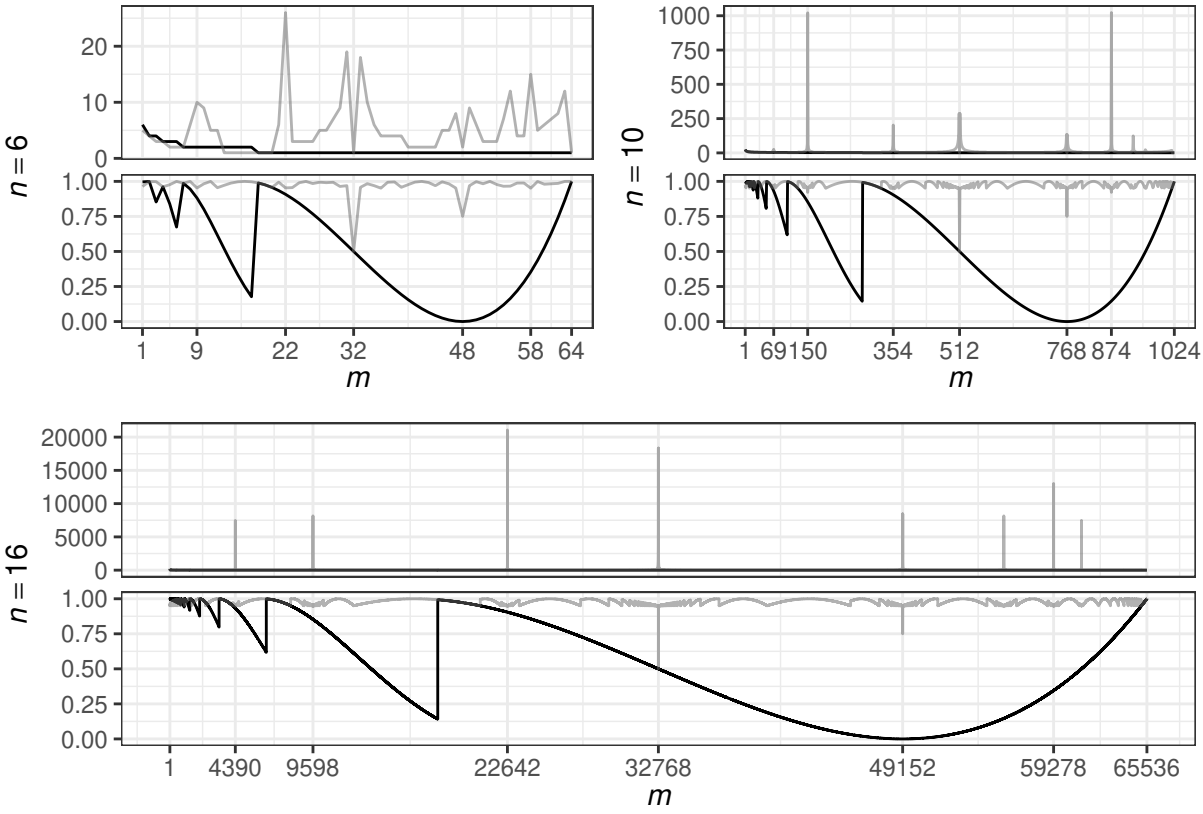
In order to determine the number of iterations required by Grover's algorithm, we use Algorithm 2 (hereafter referred to as procedure `GI2`ρ). For a system of n qubits and m target basis states, `GI2`ρ returns the minimum number of iterations required to be at least ρ percent certain to measure a target basis state. In a first step of the procedure, we initialize the current probability amplitude of the other basis states (α_o), the current probability amplitude of the target basis states (α_t), and the current iteration (i). Next, in each iteration i, we calculate the average probability amplitude ($\bar{\alpha}$) as the weighted sum of the probability amplitudes of the target and the other basis states. We use the average probability amplitude to update the probability amplitude of the target states (i.e., we reflect α_t about $\bar{\alpha}$), and calculate $P(n, m, i)$; the probability to successfully identify a target basis state after i Grover iterations in a system that has n qubits and m target basis states. If $P(n, m, i) \geq \rho$, we return i as the number of required iterations. Otherwise, if $P(n, m, i) < \rho$, we update the probability amplitude of the regular states and increment i. We continue until the required number of iterations has been identified.

Algorithm 2: Procedure (`GI2`ρ) to determine the number of iterations required by Grover's algorithm to measure a target basis state with probability of at least ρ in a system that has n qubits and m target basis states.

initialize $\alpha_o = (\sqrt{2^n})^{-1}$, $\alpha_t = -\alpha_o$, and $i = 1$;
do
  $\bar{\alpha} = 2^{-n}((2^n - m)\alpha_o + m\alpha_t)$;
  $\alpha_t = \alpha_t - 2\bar{\alpha}$;
  $P(n, m, i) = \alpha_t^2$;
  if $P(n, m, i) \geq \rho$ **then**
    return i;
  $\alpha_o = 2\bar{\alpha} - \alpha_o$;
  $i = i + 1$;
while true;

Using procedure `GI2`ρ, we perform an experiment to determine the number of Grover iterations required to measure a target basis state with 95 percent confidence for various values of n and m. Figure 12 presents the results of this experiment, and compares the performance of `GI2`ρ (with $\rho = 0.95$) and approximation $\pi 4^{-1}\sqrt{m^{-1}2^n}$ (implemented as $I(n, m)$). For each value of $n : n \in \{6, 10, 16\}$, Figure 12 shows the number of Grover iterations for both approaches (top panel) as well as the probability to measure a target basis state (bottom panel). From Figure 12, it becomes clear that approximation $\pi 4^{-1}\sqrt{m^{-1}2^n}$ cannot be used with confidence. In addition, Figure 12 also shows that, even if we use `GI2`ρ to determine the required number of Grover iterations, it may be impossible to achieve an acceptable level of confidence to measure a target basis state. For instance, if half of the basis states are target basis states, the probability to measure a target basis state always equals 0.5, regardless of the number of Grover iterations that are used (i.e., in this case, using

Figure 12. Comparison of the performance of approximation $\pi 4^{-1}\sqrt{m^{-1}2^n}$ (black line; implemented as $I(n,m)$) and procedure `GI2`ρ (gray line; with $\rho = 0.95$). For each value $n : n \in \{6, 10, 16\}$, the top panel shows the number of Grover iterations whereas the bottom panel shows the probability to measure a target basis state.



Grover's algorithm is equivalent to randomly guessing a target basis state). This illustrates that Grover's algorithm does not belong to quantum complexity class BQP (Bounded-error Quantum Polynomial time; the quantum counterpart of complexity class BBP), as it cannot guarantee that a target basis state is measured (i.e., that a solution is found) with an error probability of at most $1/3$.

When solving discrete optimization problems, these additional problems may: (1) make it impossible to find the optimal solution, and (2) result in the need for additional Grover iterations, thereby further limiting the potential speedup of quantum algorithms that rely on Grover's algorithm. In the next section, we investigate the impact of these problems on the performance of `BSP`.

8. Performance of procedure `BSP`

If we use Grover's algorithm to solve a discrete optimization problem that has n decision variables, we essentially face two problems: (1) the value of the optimal solution (V^*) is unknown, and (2)

for any given value V, the number of valid solutions (m) is also unknown. Procedure BSP addresses these problems by means of a binary search procedure that evaluates L objective values V. For each objective value V, BSP performs up to $(n+1)$ runs of Grover's algorithm to verify whether a solution exists that has a value of at least V. In each run $i : 0 \leq i \leq n$, we assume there are $\widehat{m} = n^{2-i}$ valid solutions, and perform $I(n, \widehat{m})$ Grover iterations. In the previous section, however, we have seen that approximation $I(n, \widehat{m})$ may result in a low probability to measure a valid solution (even if $\widehat{m} = m$). As such, what is the impact on BSP? Can we still rely on BSP to find the optimal solution?

Before we answer this question, note that is certainly possible to construct an example where BSP only has a small probability to find the optimal solution (e.g., we only have a 42.61 percent probability to find the optimal solution for a knapsack problem that has $n = 3$, $W = 3$, and $v_i = w_i = 1$ for all items $i : 1 \leq i \leq n$). In general, however, one can be quite confident that BSP will output the optimal solution. To explain why, we observe two facts. First, if there are m valid solutions, the probability of measuring a valid solution after $(n+1)$ runs can be obtained through the following recursive relationship:

$$\phi_i = \phi_{i-1} + (1 - \phi_{i-1}) P(n, m, I(n, 2^{n-i})), \tag{7}$$

where $\phi_{-1} = 0$. Second, the probability amplitude of the target basis states is a sine function (see also Figure 7). As a result, if we assume that $\widehat{m} = 2^n$, $\widehat{m} = 2^{(n-1)}$, $\ldots$, $\widehat{m} = 2$, $\widehat{m} = 1$ is a random sample of that sine function, $P(n, m, I(n, \widehat{m}))$ equals 0.5 on average, and hence, $\phi_n \to 1$, even for small n. This is also illustrated in Table 5. Table 5 presents ϕ_i after each of the $(n+1)$ runs of Grover's algorithm for a system that has $n = 6$ qubits, and for various values of m (the values of m that are used in the table were selected because they have a low probability to yield a valid solution, or because they require a large number of Grover iterations in order to ensure a probability of at least 0.95 to measure a valid solution; see also Figure 12). Table 5 also reports the expected total number of Grover iterations required to measure a valid solution (last column). It is clear that, regardless of the value of m, we are almost certain to measure a valid solution. In the worst case, $m = 59$, and we have to perform $(n+1) = 7$ runs of Grover's algorithm to be 97.9 percent certain to measure a valid solution (this is the smallest value of ϕ_n over all values of m). Note that, if $m = 0$, we are 100 percent certain that no valid solution will be found after performing 7 runs (requiring a total of $1+1+2+2+3+4+6 = 19$ Grover iterations). In other words, even for small n, we are almost certain to measure a valid solution (if it exists). Because procedure BSP evaluates L values V, the probability of finding the optimal solution is the product of L probabilities ϕ_n (one for each value V). Given that $\phi_n \to 1$, however, the probability to find the optimal solution also approaches 1 if n is sufficiently large. We conclude that, in general, BSP can be used with confidence.

Next, we investigate whether procedure BSP can offer a speedup when compared to classical algorithms. To solve a discrete optimization problem that has n binary decision variables and optimal objective value V^*, BSP performs L iterations of an outer loop, where L is a logarithmic

Table 5. Probability to measure one of the m valid solutions after i runs of Grover's algorithm in a system that has 6 qubits, and expected number of Grover iterations (E [GI]) required to successfully measure a valid solution, for various values of m. The worst performance (over all values of m) corresponds to the case where $m = 59$, with a probability of 0.979 to measure a valid solution after $(n + 1)$ runs; after a total of 19 Grover iterations (indicated in red).

i	0	1	2	3	4	5	6	
$\widehat{m}$	64	32	16	8	4	2	1	
$I(n, \widehat{m})$	1	1	2	2	3	4	6	
m	ϕ_i							E [GI]
0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	19.00
1	0.135	0.251	0.509	0.678	0.868	0.976	1.000	5.981
3	0.371	0.604	0.916	0.982	1.000	1.000	1.000	2.644
6	0.646	0.875	1.000	1.000	1.000	1.000	1.000	1.605
9	0.836	0.973	0.997	1.000	1.000	1.000	1.000	1.229
17	0.997	1.000	1.000	1.000	1.000	1.000	1.000	1.003
22	0.908	0.991	0.991	0.991	0.999	0.999	1.000	1.158
31	0.547	0.795	0.881	0.931	0.973	0.983	0.988	2.290
32	0.500	0.750	0.875	0.938	0.969	0.984	0.992	2.508
33	0.453	0.701	0.874	0.947	0.968	0.988	0.996	2.690
48	0.000	0.000	0.750	0.938	0.984	0.984	0.996	4.770
58	0.354	0.583	0.583	0.583	0.719	0.969	0.981	5.511
59	0.436	0.682	0.689	0.696	0.745	0.921	0.979	4.829
64	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

function of the bounds on V^*. For each iteration of the outer loop, `BSP` performs up to $(n+1)$ runs of Grover's algorithm. For each run $i : 0 \leq i \leq n$, we perform $I(n, 2^{n-i})$ Grover iterations. Each Grover iteration requires two calls to function $f_{\mathbf{x}}$. In addition, after each run of Grover's algorithm, we need to verify whether the measured solution is valid, resulting in another call to function $f_{\mathbf{x}}$. In total, `BSP` requires at most $L \sum_{i=0}^{n} \left(1 + 2I(n, 2^{n-i})\right) \approx O(L\sqrt{2^n})$ calls to $f_{\mathbf{x}}$. If we consider the binary knapsack problem, function $f_{\mathbf{x}}$ requires $O(n)$ operations, and as such, `BSP` requires at most $O(Ln\sqrt{2^n})$ operations to solve a binary knapsack problem that has n items. In practice, however, `BSP` may run much faster because not all $(n + 1)$ runs of Grover's algorithm may be required (i.e., a valid solution may be measured in run $i < (n+1)$; see also the last column of Table 5 that shows that a valid solution is often measured after only a few Grover iterations).

To verify the practical runtime of procedure `BSP`, we solve a set of 108000 instances of the binary knapsack problem. We use the instance generator of Pisinger (2005) to generate six sets of 1000 problems for each value of $n : n \in \{3, 4, \ldots, 19, 20\}$:

- Set 1 has uncorrelated values and weights that range from 1 to 1000.
- Set 2 has strongly correlated values and weights that range from 1 to 1000.
- Set 3 has weights that range from 1 to 1000 and values $v_i = 3\lceil w_i/3 \rceil$ for each item i (so-called "profit-ceiling instances").
- Set 4 has uncorrelated values and weights that range from 1 to 1000000.
- Set 5 has strongly correlated values and weights that range from 1 to 1000000.

- Set 6 has weights that range from 1 to 1000000 and values $v_i = 3\lceil w_i/3 \rceil$ for each item i.

The "hardness" of each set is determined by the type of instances, and by the range of values and weights. Instances that have uncorrelated values and weights are considered to be very easy to solve. Instances that have strongly correlated values and weights are considered to be hard to solve. The so-called "profit-ceiling" instances are considered very hard to solve. A large range of weights and values also makes instances more difficult to solve.

The results of the experiment are summarized in Table 6 and Figure 13. Table 6 reports the minimum probability to find the optimal solution for each problem set and for each value of $n : 3 \leq n \leq 20$. As expected, `BSP` can be used with confidence to solve all but the smallest binary knapsack problems. Figure 13, on the other hand, shows the expected worst-case number of function calls as well as the expected number of function calls when solving 1000 binary knapsack problems for each problem set and for each value of n. From Figure 13, we can conclude that: (1) the type of problem instance only has limited impact on the performance of the procedure (a discussion on the slight differences among the different problem sets is available in Part II of this paper), (2) the range of the weights and values has a logarithmic effect on the expected number of function calls, and (3) the expected number of function calls is much smaller than the worst-case number of function calls. To further explore the practical performance of `BSP`, we return to Table 6 that reports the worst-case number of function calls as well as the expected number of function calls divided by $L\sqrt{2^n}$. It seems that, in the worst case, `BSP` requires $\approx 5L\sqrt{2^n}$ function calls. In practice, however, only $\approx 3L\sqrt{2^n}$ calls are required. Even though this is a significant reduction in the number of function calls, this implies that, in practice, `BSP` still needs $O(nL\sqrt{2^n})$ operations when solving binary knapsack problems. The best classical algorithms, on the other hand, require $O(nW)$ and $O(n\sqrt{2^n})$ operations (Kellerer et al., 2004). As such, procedure `BSP` is able to match the performance of the best classical algorithms only if: (1) L is sufficiently small, and (2) W is sufficiently large (i.e., if $O(nW) > O(n\sqrt{2^n})$).

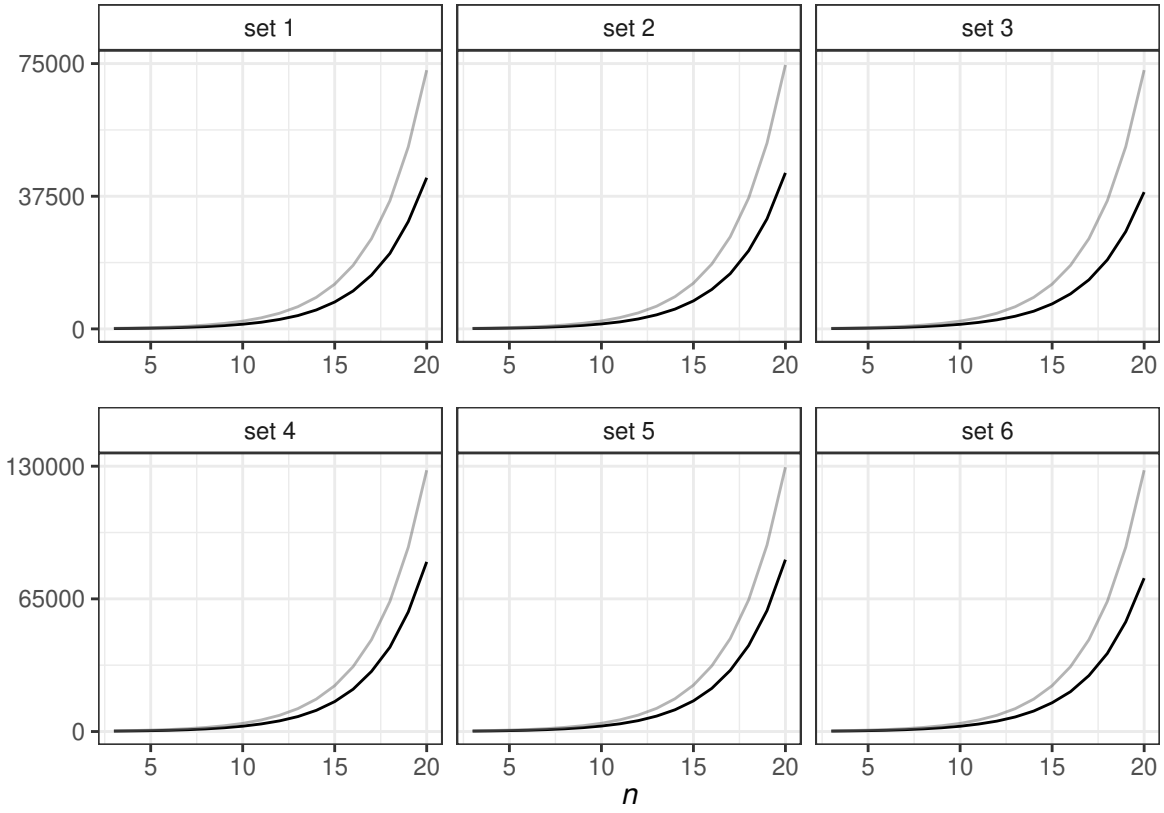
9. Discussion and conclusion

In this paper, we investigate the potential of quantum computing to solve discrete optimization problems. for this purpose, we first introduce the fundamentals of quantum computing as well as the Deutsch algorithm. Even though the Deutsch algorithm solves a problem that has no practical use, it is of particular interest because it serves as a stepping stone towards Grover's algorithm. Arguably, Grover's algorithm is the most important quantum algorithm. It can be used to perform an unstructured search, and achieves a quadratic speedup when compared to the best classical algorithm. As a search algorithm, it can be used to e.g., solve discrete optimization problems. When using Grover's algorithm to solve discrete optimization problems, however, a number of problems arise. We identify 4 problems that are often overlooked in the literature: (1) we need a search procedure that evaluates different objective values in order to determine the (unknown) optimal solution value, (2) for a given objective value V, the number of valid solutions is unknown,

Table 6. Minimum probability to find the optimal solution, average worst-case, and average expected number of function calls divided by $\sqrt{2^n}$ when using procedure `BSP` for solving 1000 binary knapsack problems for each problem set and for each value of $n : 3 \leq n \leq 20$

n	minimum probability						E $\left[\text{WCcalls}/L\sqrt{2^n}\right]$						E $\left[\text{calls}/L\sqrt{2^n}\right]$					
	set 1	set 2	set 3	set 4	set 5	set 6	set 1	set 2	set 3	set 4	set 5	set 6	set 1	set 2	set 3	set 4	set 5	set 6
3	0.91	0.93	0.93	0.93	0.93	0.95	5.41	5.40	5.41	5.53	5.53	5.53	3.13	3.35	3.24	3.33	3.43	3.40
4	0.97	0.99	0.97	0.99	0.99	0.99	5.50	5.50	5.50	5.61	5.61	5.61	3.18	3.34	3.29	3.38	3.46	3.45
5	0.98	0.98	0.98	0.98	0.98	0.98	5.40	5.44	5.41	5.52	5.54	5.52	3.13	3.32	3.27	3.39	3.50	3.47
6	0.99	0.99	0.99	0.99	0.99	0.99	5.40	5.42	5.40	5.51	5.52	5.51	3.21	3.35	3.30	3.48	3.56	3.53
7	0.99	0.99	0.99	0.99	0.99	0.99	5.45	5.42	5.45	5.54	5.53	5.54	3.29	3.38	3.32	3.54	3.61	3.58
8	0.99	1.00	1.00	0.99	0.99	0.99	5.47	5.43	5.47	5.56	5.54	5.56	3.30	3.40	3.30	3.61	3.65	3.62
9	1.00	1.00	1.00	1.00	1.00	1.00	5.41	5.42	5.41	5.52	5.52	5.52	3.28	3.39	3.23	3.60	3.65	3.58
10	1.00	1.00	1.00	1.00	1.00	1.00	5.35	5.40	5.35	5.45	5.48	5.45	3.22	3.37	3.16	3.58	3.65	3.53
11	1.00	1.00	1.00	1.00	1.00	1.00	5.34	5.42	5.34	5.44	5.48	5.44	3.20	3.36	3.12	3.57	3.62	3.52
12	1.00	1.00	1.00	1.00	1.00	1.00	5.32	5.37	5.32	5.40	5.43	5.40	3.22	3.34	3.08	3.54	3.60	3.46
13	1.00	1.00	1.00	1.00	1.00	1.00	5.31	5.30	5.32	5.38	5.39	5.38	3.19	3.29	3.04	3.51	3.59	3.43
14	1.00	1.00	1.00	1.00	1.00	1.00	5.30	5.24	5.30	5.37	5.34	5.37	3.19	3.22	2.97	3.51	3.54	3.39
15	1.00	1.00	1.00	1.00	1.00	1.00	5.27	5.19	5.27	5.34	5.30	5.34	3.16	3.18	2.94	3.49	3.50	3.35
16	1.00	1.00	1.00	1.00	1.00	1.00	5.23	5.16	5.23	5.31	5.27	5.31	3.12	3.14	2.88	3.46	3.46	3.26
17	1.00	1.00	1.00	1.00	1.00	1.00	5.18	5.15	5.18	5.29	5.25	5.29	3.09	3.09	2.82	3.46	3.45	3.22
18	1.00	1.00	1.00	1.00	1.00	1.00	5.15	5.16	5.15	5.26	5.25	5.26	3.03	3.08	2.78	3.41	3.43	3.16
19	1.00	1.00	1.00	1.00	1.00	1.00	5.14	5.18	5.14	5.24	5.26	5.24	3.03	3.07	2.74	3.40	3.41	3.12
20	1.00	1.00	1.00	1.00	1.00	1.00	5.14	5.20	5.14	5.24	5.27	5.24	3.00	3.08	2.71	3.40	3.43	3.07

Figure 13. Comparison of average worst-case number of function calls (gray) and average number of expected function calls (black) when using procedure `BSP` for solving 1000 binary knapsack problems for each problem set and for each value of $n : 3 \leq n \leq 20$.

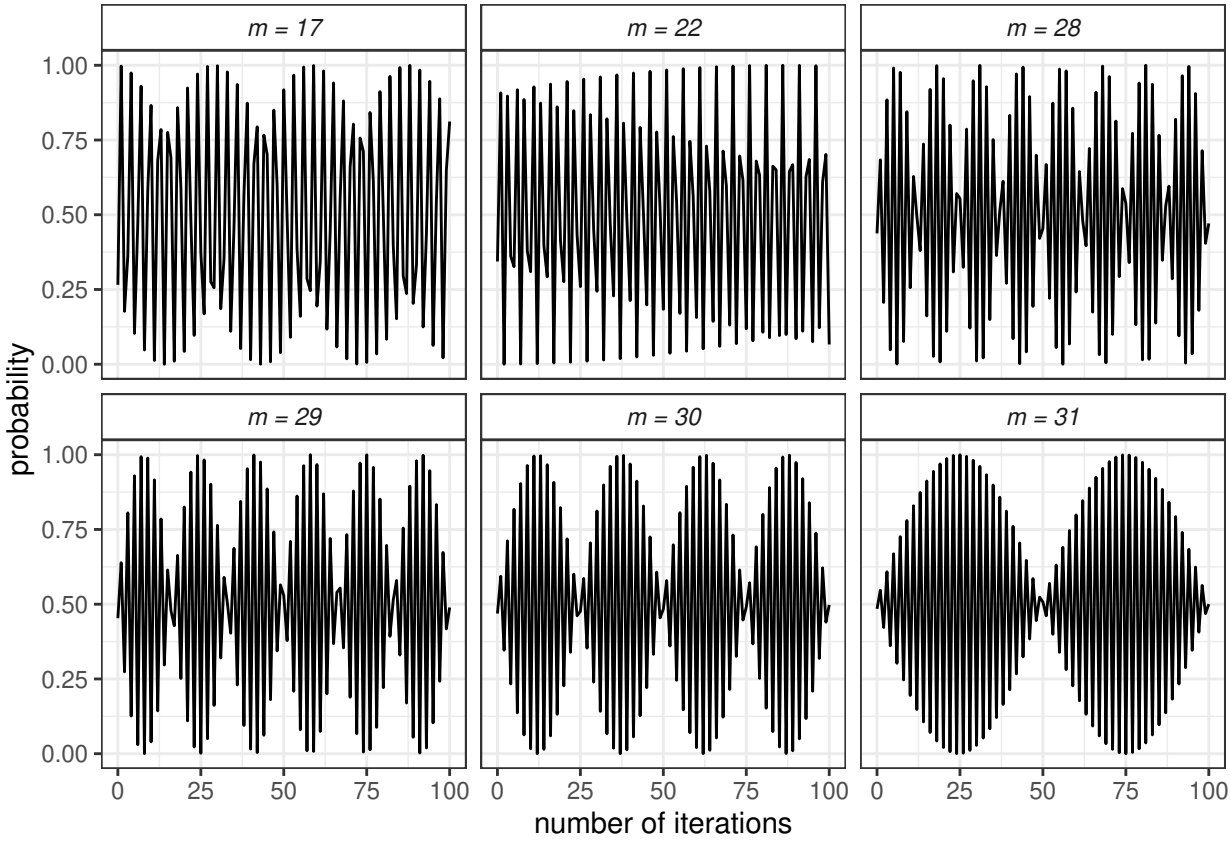


and a procedure is required that evaluates different assumed numbers of valid solutions, (3) the output of Grover's algorithm is probabilistic; there is no guarantee that the optimal solution will be found, and (4) Grover's algorithm adopts an approximation that is inaccurate; this may further decrease the probability that the optimal solution is found. To address the first two problems, we develop a procedure (`BSP`) that uses a binary search procedure that evaluates L objective values V (where L is a logarithmic function of the bounds on the optimal solution V^*). For each objective value V, procedure `BSP` performs up to $(n+1)$ runs, and for each run, Grover's algorithm is used as a subroutine to find a solution in a set of feasible solutions. With respect to the last two problems, we show that they do not impact the performance of `BSP`, and that `BSP` is almost certain to find the optimal solution.

We use procedure `BSP` to solve the binary knapsack problem, and show that it requires $O(nL\sqrt{2^n})$ operations in the worst case. We also verify the practical runtime of `BSP` by means of an extensive computational experiment in which 108000 binary knapsack problems are solved. The experiment shows that, also in practice, `BSP` requires $O(nL\sqrt{2^n})$ operations. The best classical algorithms, on the other hand, can solve the binary knapsack problem using only $O(nW)$ and $O(n\sqrt{2^n})$ operations. Therefore, even though `BSP` cannot outperform the best classical algorithms, it can match their performance if: (1) L is sufficiently small, and (2) W is sufficiently large. The computational experiment, however, also shows that the performance of `BSP` is not impacted by the type of binary knapsack problem. This highlights perhaps the biggest advantage of `BSP`: it can be used to solve any discrete optimization problem. For discrete optimization problems whose structure can be exploited by classical algorithms (e.g., the binary knapsack problem), it may not be possible to achieve a speedup by using `BSP`. If the structure of a problem is difficult to exploit, on the other hand, classical algorithms may require close to $O(2^n)$ calls to a function $f_{\mathbf{x}}$ that evaluates a solution $\mathbf{x}$, and procedure `BSP` can provide an almost quadratic speedup (assuming L is small when compared to 2^n). In addition, `BSP` can be used to solve any integer program, regardless of whether objective function and/or constraints are linear (note that `BSP` can easily be generalized to accommodate non-binary decision variables; see also Part II of this paper). The question that remains is: can we do better? Can we improve procedure `BSP` such that it requires less than $O(nL\sqrt{2^n})$ operations? Can we adapt `BSP` such that it can also exploit the structure of the problem? These are the questions that will be answered in Part II of this paper.

The contributions of Part I of this paper may be summarized as follows: (1) we identify several (often overlooked) problems that need to be overcome if we want to use Grover's algorithm to solve discrete optimization problems, (2) we develop an new algorithm that uses Grover's algorithm as a subroutine, and use this algorithm to solve a set of 108000 instances of the binary knapsack problem, (3) we are the first to effectively implement Grover's algorithm to solve a discrete optimization problem, (4) we define the conditions under which our algorithm can match the performance of the best classical algorithms when solving the binary knapsack problem, and (5) we define the conditions under which our algorithm can outperform classical algorithms in a more general setting.

Figure 14. Probability of measuring a target basis state after each Grover iteration for a system that has 6 qubits and 17, 22, 28, 29, 30, and 31 target basis states.



Appendix

Figure 14 illustrates the probability to measure any of the $m : m \in \{17, 22, 28, 29, 30, 31\}$ target basis states after each Grover iteration in a system that has $n = 6$ qubits.

References

Aaronson, S. 2018. *Introduction to Quantum Information Science*. Lecture Notes, UT Austin.

Aharonov, D. 2003. A simple proof that Toffoli and Hadamard are quantum universal, unpublished preprint at https://doi.org/10.48550/arXiv.quant-ph/0301040.

Albash, T., Lidar, D.A. 2018. Adiabatic quantum computation. *Rev. Mod. Phys.*, **90**(1), 015002.

Benioff, P. 1980. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *J. Stat. Phys.*, **22**(5), 563–591.

Born, M. 1926. Zur Quantenmechanik der Stoßvorgänge. *Z. Phys.*, **37**(12), 863–867.

Brassard, G., Høyer, P., and Tapp, A. 1998. Quantum counting. In Larsen, K.G., Skyum, S., and Winksel, G. (Eds.), *Automata, Languages and Programming*, 820–831. Springer Berlin Heidelberg, Berlin, Heidelberg.

Boyer, M., Brassard, G., Høyer, P., and Tapp, A. 1996. *Tight Bounds on Quantum Searching*. Proceedings of the Fourth Workshop on Physics and Computation, New England Complex Systems Institute.

Cheng, B., Deng, X.H., Gu, X., He, Y., Hu, G., Huang, P., Li, J., Lin, B.C., Lu, D., Lu, Y., Qiu, C., Wang, H., Xin, T., Yu, S., Yung, M.H., Zeng, J., Zhang, S., Zhong, Y., Peng, X., Nori, F., and Yu, D. 2023. Noisy intermediate-scale quantum computers. *Front. Phys.*, **18**.

Cleve, R., Ekert, A., Macchiavello, C., and Mosca, M. 1998. Quantum algorithms revisited. *Proc. R. Soc. Lond. A*, **454**(1969), 339–354.

Deutsch, D. 1985. Quantum theory, the Church–Turing principle and the universal quantum computer. *Proc. R. Soc. Lond. A*, **400**(1818), 97–117.

Deutsch, D., Jozsa, R. 1992. Rapid solution of problems by quantum computation. *Proc. R. Soc. Lond. A*, **439**(1907), 553—558.

Dirac, P. A. M. 1939. A new notation for quantum mechanics. *Math. Proc. Camb. Philos. Soc.*, **35**(3), 416–418.

Draper, T. G. 2000. Addition on a quantum computer, unpublished preprint at https://doi.org/10.48550/arXiv.quant-ph/0008033.

Feynman, R. P. 1982. Simulating physics with computers. *Int. J. Theor. Phys.*, **21**(6-7), 467–488 .

Glover, F., Kochenberger, G., and Hennig, R. 2022. Quantum bridge analytics I: a tutorial on formulating and using QUBO models. *Ann Oper Res*, **314**(1), 141–183.

Grover, L. K. 1996. A fast quantum mechanical algorithm for database search. *Proc. Annu. ACM Symp. Theory Comput.*, 212–219.

Hidary, J.D. 2021. *Quantum Computing: An Applied Approach*. Springer Nature Switzerland AG, Cham, Switzerland.

Kellerer, H., Pferschy, U., and Pisinger, D. 2004. *Knapsack Problems*. Springer Berlin Heidelberg, Berlin, Heidelberg.

MacQuarrie, E.R., Simon, C., Simmons, S., and Maine, E. 2020. The emerging commercial landscape of quantum computing. *Nature Reviews Physics*, **2**, 596–598.

McKinsey 2021. *Quantum computing: An emerging ecosystem and industry use cases*. McKinsey.

McKinsey 2023. *Quantum technology monitor*. McKinsey.

Nielsen, M., Chuang, I. 2010. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, Cambridge.

Parker, R.G., Rardin, R.L. 1988. *Discrete Optimization*. Academic Press, Inc., San Diego.

Pisinger, D. 2005. Where are the hard knapsack problems? *Comput. Oper. Res.*, **32**(9), 2271–2284.

Preskill, J. 2018. Quantum Computing in the NISQ era and beyond. *Quantum 2*, **2**, 79–99.

Shor, P.W. 1994. Algorithms for quantum computation: Discrete logarithms and factoring. *Proc. Annu. Symp. FOCS*, 124–134.

Wolsey, L.A. 2021. *Integer Programming*. Wiley, Hoboken.

Xia, H., Li, H., Zhang, H., Liang, Y., and Xin, J. 2018. An efficient design of reversible multi-bit quantum comparator via only a single ancillary bit. *Int. J. Theor. Phys.*, **57**(12), 3727–3744.

Chapter 2

Discrete Optimization a Quantum Revolution

The content of this chapter is presented in the form of a paper titled "**Discrete Optimization a Quantum Revolution (Part II)**".

2.1 Chapter 2 introduction

In this chapter, our objective is to enhance a previously achieved result by the algorithms proposed on the previous chapter and explore potential improvements. Our focus revolves around two prominent quantum algorithms: the quantum counting algorithm by Brassard et al. (1998) and the nested quantum search algorithm by Cerf et al. (2000). Despite the existence of numerous quantum algorithms, our attention is directed towards these specific algorithms due to their distinct attributes, as elaborated below.

The quantum counting algorithm proposed by Brassard et al. (1998) integrates the Quantum Phase Estimation (QPE) algorithm, analogous to Shor (1994) and Kitaev (1995), with Grover's algorithm. This amalgamation aims to approximate the number of valid solutions (m) within a set of 2^n solutions. However, recent findings by Creemers and Pérez (2023) (Chapter 3 of this thesis) reveal that Brassard et al.'s algorithm necessitates 2^n Grover iterations for an accurate

m approximation, which is equivalent to a brute-force classical approach.

Cerf et al.'s (2000) nested quantum search algorithm, akin to a classical nested search algorithm, facilitates the identification of a valid solution in a set of 2^n solutions using $O(\sqrt{2}\gamma n)$ Grover iterations, where γ is a parameter dependent on the problem instance's structure. Creemers and Pérez (2023) employ this algorithm, leading to our conclusion: (1) the (GUM) algorithm (Chapter 1) demonstrates optimal performance in assessing the existence of a valid solution in a (partial) problem, and (2) a classical nested search, augmented with GUM, surpasses its quantum counterpart.

Building on these insights, we propose two novel classical algorithms utilizing GUM to measure valid solutions for (partial) problems. The first, a hybrid branch-and-bound procedure (HBB), employs a depth-first search strategy across a tree with n levels for solving discrete optimization problems with n decision variables. Unlike HBB, the second algorithm, a random-ascent procedure (RAP), lacks the ability to exploit classical techniques. However, we demonstrate that RAP efficiently solves discrete optimization problems with n binary decision variables using $O(\eta\sqrt{2}n)$ operations, matching the performance of the best classical algorithms for the binary knapsack problem.

To evaluate HBB and RAP, we apply both procedures to solve 66,000 instances of the binary knapsack problem. Initial results indicate that both procedures can solve problems with n items using $O(n\sqrt{2}n)$ operations. Further enhancements include guiding each run of GUM using previous information, employing a heuristic lower bound procedure, and generalizing GUM. While HBB dominates RAP when exploiting problem structure, a new combined procedure, RAHBB, is introduced to leverage the strengths of both.

Expanding beyond binary decision variables, we demonstrate the versatility of HBB and RAP in solving problems with n discrete decision variables using $O(\eta\sqrt{2}nb)$ operations, where b is the number of discrete values assignable to a decision variable. Not confined to well-known problems, such as the binary knapsack problem, HBB and RAP emerge as general-purpose solvers capable of addressing complex, non-linear integer programming problems. This positions them as state-of-the-art quantum procedures with the potential to revolutionize the field of discrete optimization.

Discrete optimization: A quantum revolution (Part II)

Stefan Creemers*
IESEG School of Management, Lille, France, s.creemers@ieseg.fr
KU Leuven, Leuven, Belgium, stefan.creemers@kuleuven.be

Pérez Luis Fernando
IESEG School of Management, Lille, France, l.perezarmas@ieseg.fr

In part I of this paper, we introduce the basics of quantum computing, and use Grover's algorithm as a subroutine in a binary search procedure that can solve any discrete optimization problem. In part II, we improve the performance of this binary search procedure. For this purpose, we propose two new procedures, and use them to solve 66000 instances of the binary knapsack problem. Our results show that both procedures can match the performance of the best classical algorithms. Whereas the first procedure (a hybrid branch-and-bound algorithm) can exploit the structure of the problem, the second procedure (a random-ascent algorithm) can be used to solve discrete optimization problems that have no clear structure and/or are difficult to solve using traditional methods. After improving and generalizing these procedures, we show that they can solve any discrete optimization problem using at most $O(\eta\sqrt{2^{nb}})$ operations, where η is the number of operations required to evaluate the feasibility of a solution, n is the number of decision variables, and b is the number of discrete values that can be assigned to each decision variable. In addition, we show that both procedures can also be used as a heuristic to find (near-)optimal solutions using far less than $O(\eta\sqrt{2^{nb}})$ operations.

Keywords: Quantum; computing; algorithm; knapsack.

1. Introduction

In Part I of this paper, we introduce a binary search procedure (hereafter referred to as BSP) that uses Grover's algorithm to solve any discrete optimization problem that has n binary decision variables using $O(L\eta\sqrt{2^n})$ operations, where (1) L is a logarithmic function of the difference between upper and lower bounds on optimal solution value V^*, and (2) η is the number of operations required by a function $f_{\mathbf{x}}$ that returns 1 if $\mathbf{x}$ is a valid solution that satisfies the constraints of the optimization problem (and 0 otherwise). We show that BSP is able to overcome several (often overlooked) problems that arise when using Grover's algorithm, and perform a computational experiment in which we solve 108000 instances of the binary knapsack problem. For problems that have n items, our results show that BSP can find the optimal solution using $O(Ln\sqrt{2^n})$ operations. Therefore, if L is sufficiently small, BSP can compete with the best classical algorithms (that require $O(n\sqrt{2^n})$ operations; see e.g., Kellerer et al. (2004)).

*Corresponding author

An outline of BSP is presented in Algorithm 1. To keep things simple, we assume that BSP solves a discrete optimization problem that has n binary decision variables, and that maximizes an integer objective value V (note, however, that BSP can easily be adapted to solve more general discrete optimization problems). BSP uses a binary search procedure that performs L iterations in which we update either lower bound $V_{\min}$ or upper bound $V_{\max}$. In each iteration, BSP calls Algorithm 2 (hereafter referred to as GUM) to assess whether a valid solution exists that satisfies the constraints of the optimization problem, and that has a value of at least $V = \lfloor 0.5(V_{\min} + V_{\max}) \rfloor$. If a valid solution $\mathbf{x}$ is found, we update $V_{\min}$, and let $\mathbf{x}^* = \mathbf{x}$ and $V^* = V$ (i.e., the last found valid solution is also an optimal solution). If no valid solution can be found, we update $V_{\max}$. This process continues until $V_{\min} > V_{\max}$. To assess whether a valid solution exists, GUM uses Grover's algorithm. Grover's algorithm performs $I(n, m) = \left\lceil \pi 4^{-1} \sqrt{m^{-1} 2^n} \right\rceil$ Grover iterations to maximize the probability to find any of the m valid solutions in a set of 2^n solutions. Often, however, m is unknown (i.e., we don't know the number of valid solutions that have a value of at least V). Therefore, GUM evaluates several assumed numbers of valid solutions ($\widehat{m}$). Starting with $\widehat{m} = 2^n$, GUM calls Grover's algorithm for each value of $\widehat{m}$, and performs $I(n, \widehat{m})$ Grover iterations. In each Grover iteration, function $f_{\mathbf{x}}$ is called twice (once to determine whether solution $\mathbf{x}$ is valid and once to "uncompute"; see also Part I of this paper). If, after $I(n, \widehat{m})$ Grover iterations, a valid solution $\mathbf{x}$ is found, GUM stops and returns $\mathbf{x}$. If no valid solution is found, we let $\widehat{m} = \widehat{m}/2$ and try again. If no solution is found after processing $\widehat{m} = 1$, GUM stops, and we conclude that no valid solution exists that has a value of at least V.

The expected number of function calls required by GUM is given by:

$$\sum_{i=0}^{n} P(n, m, 2^{(n-i)}) \left((i+1) \sum_{j=0}^{i} 2I(n, 2^{(n-i)}) \right), \tag{1}$$

where $P(n, m, \widehat{m})$ is the probability to measure any of the m valid solution in we use $I(n, \widehat{m})$ Grover iterations to search a set of 2^n solutions. In the worst case, no valid solution can be found, and GUM requires $O(\sqrt{2^n})$ function calls. If, however, there are m valid solutions, GUM requires $O(\sqrt{m^{-1} 2^n})$ function calls (see also Appendix for a discussion).

In contrast to algorithms that are dedicated to solving e.g., the binary knapsack problem, BSP can easily be adapted to solve any discrete optimization problem. As such, BSP can be seen as a general-purpose solver. In addition, given n binary decision variables, BSP only requires $O(L\eta\sqrt{2^n})$ operations. In this paper, we verify whether we can improve this result. For this purpose, we briefly discuss two well-known quantum algorithms: the quantum counting algorithm of Brassard et al. (1998) and the nested quantum search algorithm of Cerf et al. (2000). Even though there are many other quantum algorithms, these algorithms are of particular interest as is explained below.

The quantum counting algorithm of Brassard et al. (1998) combines the Quantum Phase Estimation (QPE; see also Shor (1994) and Kitaev (1995)) algorithm and Grover's algorithm to approximate the number of valid solutions (i.e., m) in a set of 2^n solutions. Unfortunately, however, Creemers and Pérez (2023) have shown that the quantum counting algorithm of Brassard et al. requires 2^n Grover iterations to accurately approximate m (i.e., this is equivalent to using a brute-force classical approach to find m). Even though we should not use the quantum counting algorithm of Brassard et al. to approximate m, it can still be used to

Algorithm 1: Binary search procedure (BSP) that was used to solve the binary knapsack problem in Part I of this paper.

initialize $\mathbf{x}^* = \emptyset$, $V^* = 0$, $V_{\min}$, $V_{\max}$, and start procedure $\texttt{BSP}(n, V_{\min}, V_{\max})$;
procedure $\texttt{BSP}(n, V_{\min}, V_{\max})$
  do
    $V = \lfloor 0.5(V_{\min} + V_{\max}) \rfloor$;
    $\mathbf{x} = \texttt{GUM}(n)$;
    if $\mathbf{x} \neq \emptyset$ **then**
      update optimal solution $\mathbf{x}^* = \mathbf{x}$ and its value $V^* = V$;
      $V_{\min} = V + 1$;
      break;
    else
      $V_{\max} = V - 1$;
      break;
  while $V_{\min} \leq V_{\max}$;

Algorithm 2: Procedure that uses Grover's algorithm to find a valid solution when the number of valid solutions (m) is unknown in a system that has n binary decision variables.

procedure $\texttt{GUM}(n)$
  $\widehat{m} = 2^n$;
  do
    perform $I(n, \widehat{m})$ iterations of Grover's algorithm equipped with $f_{\mathbf{x}}$;
    measure basis state $|\mathbf{x}\rangle$;
    if $f_{\mathbf{x}} = 1$ **then**
      return $\mathbf{x}$;
    else if $\widehat{m} > 1$ **then**
      $\widehat{m} = \widehat{m}/2$;
    else
      return $\emptyset$;
  while true;

assess whether a valid solution exists (i.e., to verify whether $m > 0$). For instance, instead of using GUM in BSP to determine whether a valid solution can be found that has a value of at least V, we can use the quantum counting algorithm of Brassard et al. to verify whether $m > 0$. This, however, also does not seem a viable strategy, as Creemers and Pérez (2023) show that GUM not only dominates any quantum counting algorithm when verifying whether $m > 0$, it also has optimal performance.

The nested quantum search algorithm of Cerf et al. (2000) is the quantum equivalent of a classical nested search algorithm, and allows to find a valid solution in a set of 2^n solutions using only $O(\sqrt{2^{\gamma n}})$ Grover iterations, where γ is some number less than or equal to 1 that depends on the structure of the problem instance. Creemers and Pérez (2023) use the nested quantum search algorithm of Cerf et al. to solve the binary knapsack problem, and find that: (1) there may only be a low probability to find the optimal solution, and (2) it may not be possible to impose all constraints of the optimization problem when performing a nested search. Unfortunately, Creemers and Pérez (2023) also show that these problems cannot be resolved, and that a nested quantum search is dominated by a classical nested search that uses e.g., GUM to find (partial) solutions to (nested) problems.

The analysis of the quantum counting algorithm of Brassard et al. and the nested quantum search algorithm of Cerf et al. allows us to conclude that: (1) GUM has optimal performance when assessing whether a (partial) problem has a valid solution, and (2) a classical nested search equipped with GUM outperforms a quantum nested search. We combine these insights to propose two new classical algorithms that use GUM to measure valid solutions to (partial) problems. Whereas the first algorithm is a hybrid branch-and-bound procedure, the second algorithm may be seen as a random-ascent procedure (in what follows, these procedures are referred to as HBB and RAP, respectively.)

To solve a discrete optimization problem that has n decision variables, HBB uses a depth-first search strategy to explore a tree that has n levels. At level $i : 1 \leq i \leq n$, we visit nodes that represent partial solutions where values have been assigned to decision variables $j < i$. If we are currently in a node at level i, and x_i is the (valid) value that has been assigned to decision variable i, HBB uses GUM to determine whether we can find a valid assignment for decision variables $j > i$ such that the best solution that was found so far can still be improved. Only if such a solution can be found, HBB explores the branch where decision variable i is assigned value x_i. Because Grover's algorithm (and hence GUM) is less accurate when finding solutions to smaller problems (see also Part I of this paper), we propose a hybrid approach, and use a classical algorithm to solve partial problems at level $i : (n - i) < \omega$; to solve problems that consider ω decision variables or less. In addition, note that HBB can exploit classical techniques to improve the performance of a branch-and-bound procedure (e.g., dominance rules and/or bounding procedures). In contrast, RAP, cannot exploit classical techniques, and is a far more simple procedure that iteratively uses GUM to draw a "random" solution from the set of solutions that can still improve the best-found solution. Even though simple, we show that RAP expects to solve any discrete optimization problem that has n binary decision variables using at most $O(\eta\sqrt{2^n})$ operations (i.e., this implies that RAP can match the performance of the best classical algorithms that solve the binary knapsack problem).

To investigate the performance of HBB and RAP, we use both procedures to solve 66000 instances of the binary knapsack problem (i.e., a subset of the aforementioned 108000 instances that were used in Part I of

Table 1. Data for example knapsack problem with three items.

i	w_i	v_i
1	2	3
2	3	1
3	2	2
$W = 4$		

this paper). The initial results show that both procedures are able to solve a binary knapsack problem that has n items using $O(n\sqrt{2^n})$ operations. To further improve the performance of both procedures, we: (1) guide each run of GUM using information of the previous run of GUM, (2) use a heuristic lower bound procedure to set an initial solution, and (3) generalize GUM. In addition, we also use an upper bound procedure to fathom nodes in HBB. We discuss the impact of these improvements, and conclude that HBB dominates RAP if it is able to exploit the structure of the problem. To exploit the strengths of both procedures, we also propose a new procedure that combines HBB and RAP. This new procedure (hereafter referred to as RAHBB) can be tailored depending on our ability to exploit the structure of the problem. Last but not least, we show that HBB and RAP can be used as heuristic procedures that are able to find (near-) optimal solutions using far less than $O(n\sqrt{2^n})$ operations.

We show how to generalize HBB and RAP (or a combination thereof) such that they can be used to solve problems that have n discrete (rather than binary) decision variables using $O(\eta\sqrt{2^{nb}})$ operations, where b is the number of discrete values that can be assigned to a decision variable. Note that, HBB and RAP are not limited to solving well-known problems for which dedicated algorithms exist (e.g., the binary knapsack problem). HBB and RAP can be seen as general-purpose solvers that can solve any discrete optimization problem, including complex, non-linear integer programming problems. As such, we argue that HBB and RAP are the state-of-the-art quantum procedures that can revolutionize the field of discrete optimization.

In what follows, procedures HBB and RAP are introduced in Sections 2 and 3. Sections 4, 5, and 6 investigate how to improve, combine, and generalize HBB and RAP. Section 7 concludes. Note that, the dynamics of HBB and RAP are illustrated using the example binary knapsack problem that was also used in Part I of this paper. The parameters of this example are provided in Table 1, where w_i and v_i denote the weight and value of item i, and W is the maximum weight capacity of the knapsack (the optimal solution of the example knapsack problem is $\mathbf{x}^* = \{1, 0, 1\}$, and has value $V^* = 5$).

2. Hybrid branch-and-bound procedure

The hybrid branch-and-bound procedure is presented in Algorithm 3 (hereafter referred to as HBB). In what follows, we use HBB to solve the binary knapsack problem (note, however, that HBB can easily be adapted to solve other discrete optimization problems). Procedure HBB uses a recursive depth-first strategy to search a tree that has n levels, and at each level $i : 1 \leq i \leq n$, we verify whether item i can be added to the current (partial) solution $\mathbf{x} = \mathbf{x}_{1(i-1)} = \{x_1, \ldots, x_{(i-1)}\}$ (that has value $V = \sum_{j=1}^{(i-1)} x_j v_j$ and remaining weight capacity $R = W - \sum_{j=1}^{(i-1)} x_j w_j$). If the weight of item i does not exceed the remaining weight capacity, we

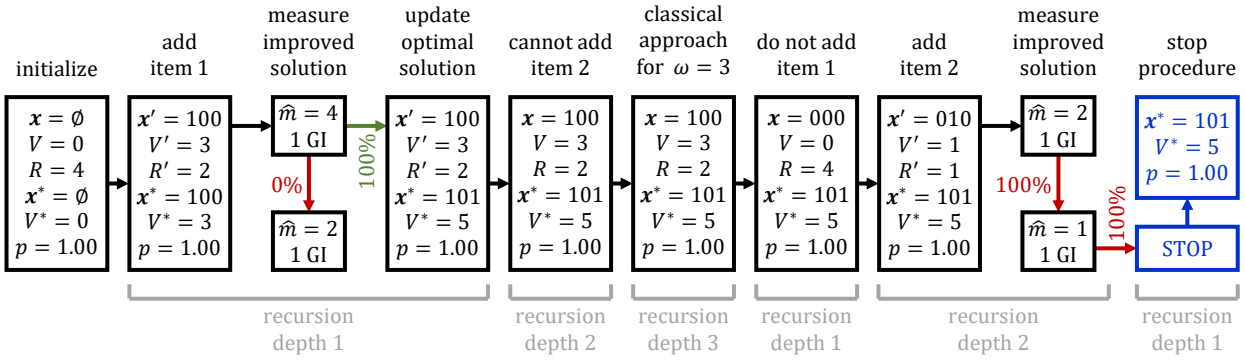
add item i and let $\mathbf{x}' = \mathbf{x} +\!\!+ \{1\}$, $V' = V + v_i$, and $R' = R - w_i$ (note that "$+\!\!+$" represents the concatenation operator, and that $\mathbf{x}_{1(i-1)} +\!\!+ x_i = \mathbf{x}_{1i} = \{x_1, \ldots, x_{(i-1)}, x_i\}$). In addition, if $V' > V^*$, we also update the best-found solution and its value. Next, we use GUM to assess whether a better solution can still be found at a lower level of the search tree. If such a solution $\mathbf{x}_{(i+1)n}$ can be found, we update the best-found solution, and enter a new recursion in which item i is added. After (potentially) exploring the branch in which we add item i, we also explore the branch in which we do not add item i. At a given level of the tree, ω items remain to be added. For small ω, however, it is possible that GUM (that relies on Grover's algorithm) is unable to measure a valid solution (even if that solution exists; see also Part I of this paper). In addition, for small ω, the quadratic speedup (and hence the computational gain) obtained by Grover's algorithm is limited. Therefore, we propose a "hybrid" procedure that uses: (1) GUM if more than ω items remain, and (2) a classical approach if ω items remain. If ω items remain, the classical approach determines the set of remaining items in $\mathcal{X}_{in}$ (i.e., the set of all partial solutions that consider items $j \geq i$) that best improves the current (partial) solution without violating the maximum weight constraint. If such a solution ($\mathbf{x}^\star_{in}$; with value $V^\star_{in}$ and weight $W^\star_{in}$) can be found, we update the best-found solution. A hybrid approach has already been proposed by e.g., Ambainis et al. (2018) who describe an algorithm that is dedicated to solving the travelling salesman problem. Procedure HBB is also similar to the backtracking algorithm of Montanaro (2016), however, the procedure of Montanaro is not hybrid, and uses a quantum counting algorithm to determine whether a valid solution exists at a deeper level of the tree. As has been shown by Creemers and Pérez (2023), quantum counting algorithms are dominated by GUM when verifying whether a valid solution exists.

Algorithm 3: Hybrid branch-and-bound procedure to solve the binary knapsack problem using GUM.

initialize $\mathbf{x}^* = \emptyset$, $V^* = 0$, and start procedure HBB$(\emptyset, 0, W, 1)$;
Procedure HBB$(\mathbf{x}, V, R, i)$
  if $n - i < \omega$ **then**
    use classical algorithm to find $\mathbf{x}^\star_{in} \in \mathcal{X}_{in}$ that has highest value $V^\star_{in} > V^* - V$ and $W^\star_{in} \leq R$;
    if $\mathbf{x}^\star_{in} \neq \emptyset$ **then**
      update best-found solution $\mathbf{x}^* = \mathbf{x} +\!\!+ \mathbf{x}^\star_{in}$ and its value $V^* = V + V^\star_{in}$;
  else
    if $w_i \leq R$ **then**
      update current solution $\mathbf{x}' = \mathbf{x} +\!\!+ \{1\}$, its value $V' = V + v_i$, and its remaining weight $R' = W - w_i$;
      Use GUM$(n - i)$ to measure $\mathbf{x}_{(i+1)n} \in \mathcal{X}_{(i+1)n}$ that has value $V_{(i+1)n} > V^* - V'$ and $W_{(i+1)n} \leq R'$;
      if $\mathbf{x}_{(i+1)n} \neq \emptyset$ **then**
        update best-found solution $\mathbf{x}^* = \mathbf{x}' +\!\!+ \mathbf{x}_{(i+1)n}$ and its value $V^* = V' + V_{(i+1)n}$;
        HBB$(\mathbf{x}', V', R', i + 1)$;
    HBB$(\mathbf{x}, V, R, i + 1)$;

To further illustrate the dynamics of HBB, we use HBB to solve the example binary knapsack problem. The steps of the procedure are illustrated in Figure 1. In this example, we assume $\omega = 1$. To initialize the

Figure 1. Illustration of the dynamics of procedure HBB when solving the example binary knapsack problem.



procedure, we let $\mathbf{x}^* = \emptyset$, $V^* = 0$, and enter a first recursion with current solution $\mathbf{x} = \emptyset$, value $V = 0$, remaining weight $R = W = 4$, and $i = 1$. In this first recursion, a classical procedure should not be used, and sufficient weight remains for item $i = 1$ to be added (i.e., $w_1 \leq R$). As a result, we update the current and best-found solution, and let $\mathbf{x}^* = \mathbf{x}' = \{1\}$, $V' = V^* = 3$, and $R' = 2$. Next, we use GUM to measure a solution $\mathbf{x}_{(i+1)n} \in \mathcal{X}_{(i+1)n}$ that has a value $V_{(i+1)n}$ larger than $V^* - V'$ (i.e., a value larger than 0) and a weight $W_{(i+1)n}$ that does not exceed R'. After adding item 1, two items are left, and $\mathbf{x}_{(i+1)n} = \{0, 1\}$ is the only valid solution (i.e., $\mathbf{x}^*$ can only be improved if we add item 3). In a system that has two items (i.e., qubits) and only one valid solution, GUM has a 100 percent probability to measure the valid solution after performing $I(2, 4) = 1$ Grover iteration. As a result, after calling GUM, we update the best-found solution, and let $\mathbf{x}^* = \{1\} + \{0, 1\} = \{1, 0, 1\}$ and $V^* = 3 + 2 = 5$. After updating the best-found solution, we enter a second recursion. In this recursion, $(n - i)$ is still smaller than ω, and as such, we do not use a classical procedure to try and improve the current solution. We are, however, also unable to add item 2 to the existing solution (that already contains item 1; $w_2 > R$). As a result, we enter a new recursion without adding item 2. In this recursion, $(n-i) < \omega$, and we use a classical procedure to determine whether a partial solution can still be found that allows to improve the best-found solution. As the optimal solution has already been found, no improvement is possible, and we exit this recursion. After ascending to recursion depth 2, we again ascend one level, and end up at recursion level 1, in which we enter a new recursion where item 1 is not added. In this recursion, the current value equals $V = 0$ and the remaining weight capacity is $R = 4$. As a result, item 2 can be added, we update the current solution, and we use GUM to measure a solution $\mathbf{x}_{(i+1)n} \in \mathcal{X}_{(i+1))n}$ that has a value bigger than $V^* - V' = 4$. No such solution exists, and therefore, after performing two Grover iterations (one for $\hat{m} = 2$ and one for $\hat{m} = 1$), we complete the procedure with a 100 percent probability to find the optimal solution.

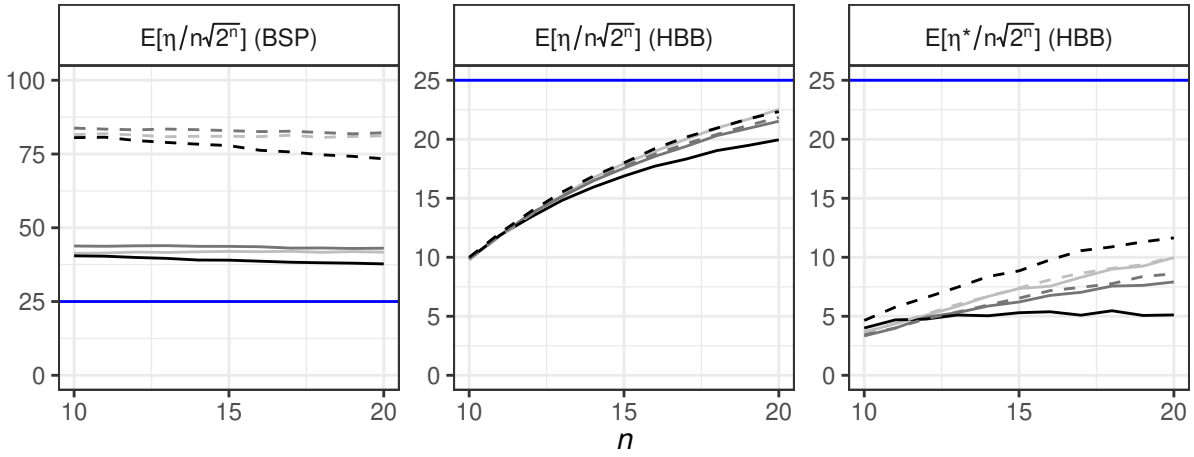
We assess the performance of HBB using a computational experiment in which 66000 instances of the binary knapsack problem are solved. For each value of $n : n \in \{10, 11, \ldots, 19, 20\}$, we use the the instance generator of Pisinger (2005) to generate six sets of 1000 problems (see also Part I of this paper):

- Set 1 has uncorrelated values and weights that range from 1 to 1000.
- Set 2 has correlated values and weights that range from 1 to 1000.

- Set 3 are so-called "profit-ceiling" instances that have weights that range from 1 to 1000 and values $v_i = 3\lceil w_i/3 \rceil$ for each item i.
- Set 4 has uncorrelated values and weights that range from 1 to 1000000.
- Set 5 has correlated values and weights that range from 1 to 1000000.
- Set 6 are profit-ceiling instances that have weights that range from 1 to 1000000.

In general, profit-ceiling instances are more difficult to solve than instances with correlation that, in turn, are more difficult to solve than instances without correlation. In addition, an increased range of values and/or weights also makes instances more difficult to solve. As there is no quantum computer that is capable of running advanced quantum algorithms, we simulate the execution of HBB 1000 times for each of the problem instances (see also Appendix for a discussion on how to assess the performance of quantum algorithms). In the experiment, we let $\omega = 5$ (i.e., if 5 items remain, a classical approach is used to determine which partial solution best improves the current solution; a value $\omega = 5$ is chosen because Grover's algorithm is sufficiently accurate when at least 6 items are considered). Note that, in the discussion of the results that follows, we omit the (insignificant) number of operations required by the classical approach. The results of the experiment are presented in Figure 2. Figure 2 shows the expected number operations divided by $n\sqrt{2^n}$ (denoted $E[\eta/n\sqrt{2^n}]$; note that we divide by $n\sqrt{2^n}$ to show that HBB requires $O(cn\sqrt{2^n})$ operations, where c is a constant) when solving the binary knapsack problems using HBB (center panel). For comparison, we also include the performance of BSP (left panel; see also Part I of this paper). It is obvious that HBB significantly outperforms BSP. In addition, it seems that HBB can solve instances that have a larger range of values (i.e., sets 4, 5, and 6) without extra effort (this in contrast to BSP; because BSP uses a binary search procedure to identify the optimal value, a larger range of values results in more operations). For both procedures, however, we observe (slight) differences in the required number of operations to solve instances of different sets. To further explore these differences, we refer to Table 2 that reports the average number of times a feasible solution value occurs over all instances in a set. For instance, for set 3, Table 2 shows that a feasible solution value occurs on average 252.58 times if $n = 20$ (i.e., given a feasible solution value V, we expect to find 252.58 feasible solutions that have the same value V). In contrast, for set 4, a feasible solution value occurs only 1.06 times if $n = 20$. If more solutions have a particular value V, more solutions can be removed from the search space if GUM can show that a solution exists that has a value higher than V (i.e., it is easier to find the best solution in a set of solutions that have 2 values that occur 500 times than in a set of solutions that have 1000 values that only occur 1 time). As a result, for BSP we see small differences between sets 1 and 2, and between sets 4, 5, and 6. For HBB, that is not impacted by an increase in the range of values, we see almost no difference between sets 1 and 4 (instances that have uncorrelated weights/values), and between sets 2 and 5 (instances that have correlated weights/values). There is however, a slight difference between instances that have uncorrelated weights/values (sets 1 and 4) and instances that have correlated weights/values (sets 2 and 5). This illustrates that, rather than the range of values, the structure of the instance can have an impact on the performance of HBB. We also observe that uncorrelated/correlated instances (sets 1, 2, 4, and 5) differ from profit-ceiling instances that are both the most easy to solve (set 3) and the most difficult to solve (set 6). Here again, the

Figure 2. Expected number of operations divided by $n\sqrt{2^n}$ required to solve a binary knapsack problem using procedures BSP and HBB, as well as the expected number of operations divided by $n\sqrt{2^n}$ required to find an optimal solution for the first time using HBB, when solving 1000 binary knapsack problems for each problem set and for each value of $n : 10 \leq n \leq 20$ (lines of sets 1 and 4 are light gray, sets 2 and 5 are dark gray, sets 3 and 6 are black, sets 1, 2, and 3 are full, and sets 4, 5, 6 are dashed). The blue line indicates a value of 25 on the vertical axis.



instances of set 3 are easiest to solve because they have a significantly higher frequency of feasible solution values. Set 6, on the other hand, may be the hardest to solve because the branch-and-bound procedure is less capable of exploiting the structure of profit-ceiling instances (when compared to uncorrelated/correlated instances, profit-ceiling instances have items that have a ratio of value over weight that is always close to 1). Notwithstanding the differences among sets, it appears that, for all sets, the expected number of operations (divided by $n\sqrt{2^n}$) converges to some constant (note that this also has been verified for some sets for values of n larger than 20). Therefore, we conclude that HBB has expected runtime $O(n\sqrt{2^n})$, which is equivalent to the performance of the best classical algorithms that solve the binary knapsack problem. Last but not least, Figure 2 also reports the expected number of operations required to find an optimal solution for the first time (denoted $E[\eta^*/n\sqrt{2^n}]$; right panel). We observe that HBB can find an optimal solution for the first time in $O(n\sqrt{2^n})$ operations. The difference between the right and center panel, however, also shows that, once an optimal solution has been found, a significant amount of time is spent to prove that this solution is optimal (at different nodes in the search tree, GUM needs to evaluate all values of $\hat{m}$ in order to be sure that no valid solution can be found that improves the optimal solution).

3. Random-ascent procedure

In the previous section, we presented a hybrid branch-and-bound procedure (HBB) that is able to solve a binary knapsack problem that has n items using only $O(n\sqrt{2^n})$ operations. Most of these operations, however, are spent on proving that the optimal solution cannot be improved. To minimize the time spent on proving that a solution is optimal, we propose another quantum algorithm. This simple, yet elegant, procedure requires only one call to GUM once the optimal solution has been found. The procedure can be seen as a random-ascent procedure that achieves a random improvement of the best-found solution value in each iteration. In

Table 2. Average number of times a feasible solution value occurs for each problem set and for each value of n : $10 \leq n \leq 20$

n	set 1	set 2	set 3	set 4	set 5	set 6
10	1.10	1.10	1.41	1.00	1.00	1.00
11	1.19	1.20	1.81	1.00	1.00	1.00
12	1.38	1.39	2.61	1.00	1.00	1.00
13	1.75	1.78	4.13	1.00	1.00	1.00
14	2.49	2.57	6.96	1.00	1.00	1.00
15	3.92	4.10	12.17	1.00	1.00	1.01
16	6.60	6.95	21.78	1.00	1.01	1.02
17	11.54	12.20	39.62	1.01	1.01	1.03
18	20.69	21.92	72.88	1.02	1.02	1.06
19	37.72	40.03	135.20	1.03	1.03	1.12
20	69.57	73.95	252.58	1.06	1.07	1.25

this section, we use the procedure to solve the binary knapsack problem (note, however, that it can easily be adapted to solve other discrete optimization problems). Algorithm 4 (hereafter referred to as RAP) provides an outline of the procedure. After initializing $x^* = \emptyset$ and $V^* = 0$, RAP enters a first iteration in which GUM is used to measure a solution that has a value $V' > V^*$ and weight $W' \leq W$. If such a solution is found, we update the best-found solution, and start a new iteration. This process continues until the optimal solution has been found. If we are lucky, we find the optimal solution after one iteration. If we're unlucky, it may take $S \leq 2^n$ iterations (where S is the number of unique feasible solutions). In general, however, we expect to find the optimal solution after $(\lceil \log_2(S) \rceil + 1)$ iterations. In addition, in the first iterations, we may expect GUM to find a solution rather quickly (as there are still many solutions at that point). In later iterations (if few solutions remain), GUM has to evaluate more (increasingly smaller) values of $\hat{m}$ that require (increasingly larger) number of Grover iterations. If, eventually, the optimal solution has been found, GUM is called one last time to verify that no solution can be found that still improves the optimal solution.

Algorithm 4: Random-ascent procedure to solve the binary knapsack problem using GUM.

initialize $\mathbf{x}^* = \emptyset$, $V^* = 0$, and start procedure RAP(n);
Procedure RAP(n)
  do
    Use GUM(n) to measure $\mathbf{x}'$ that has value $V' > V^*$ and $W' \leq W$;
    if $\mathbf{x}' \neq \emptyset$ **then**
      update best-found solution $\mathbf{x}^* = \mathbf{x}'$ and its value $V^* = V'$;
  while $\mathbf{x}' \neq \emptyset$;

To further illustrate the dynamics of RAP, we use it to solve the example binary knapsack problem. The steps of the procedure are illustrated in Figure 3. After initializing, we use GUM to measure any of the four feasible solutions that have a value $V > 0$. There is a 6.25 percent probability that no valid solution was measured (leading us to conclude that there is no feasible solution). There is, however, also a 93.75 percent probability that a valid solution was measured. In this case, there are 4 options. First, we may measure the optimal solution itself (with a probability of 25 percent). After measuring the optimal solution, we perform

one more run of GUM to show that no solution can still be found that improves the optimal solution. Second, we may measure solution $\mathbf{x} = \{1, 0, 0\}$ that has $V = 3$. In this case, we perform another iteration in which we run GUM again, and have a 99.99 percent probability to measure the only solution that can still improve a value of $V = 3$ (i.e., the optimal solution). After measuring the optimal solution, we perform one more run of GUM to verify that the optimal solution has been found. Third, we may measure solution $\mathbf{x} = \{0, 0, 1\}$ that has $V = 2$. In this case, we have a 100 percent probability to measure one of two solutions that have a value $V > 2$. If the optimal solution is measured, we run GUM to show that no more improvement can be made. If, on the other hand, we measure solution $\mathbf{x} = \{1, 0, 0\}$, we need to run GUM again to measure the optimal solution and once more to prove that there is no solution that can still improve the optimal solution. Fourth, we may measure solution $\mathbf{x} = \{0, 0, 1\}$ that has $V = 1$. In this case all of the aforementioned scenarios are possible after running GUM again. In the end, RAP has a 93.20 percent probability to measure the optimal solution (see also the blue boxes in Figure 3).

In the worst case, there are $S = 2^n$ unique feasible solutions, and we expect to perform $\lceil \log_2(S) \rceil + 1 = (n+1)$ iterations. In each iteration $i : 0 \leq i \leq n$, we expect that $m = {}^{2^n}/_{2^i} = 2^{(n-i)}$ feasible solutions remain, and that GUM performs $\sum_{j=0}^{n} \left(1 - \sum_{j'=0}^{j} P(n, 2^{(n-i)}, 2^{(n-j')})\right) P(n, 2^{(n-i)}, 2^{(n-j)}) I(n, 2^{(n-j)})$ Grover iterations to measure a valid solution. We expect to find the optimal solution after $(n+1)$ iterations, after which we call GUM one last time to prove that the optimal solution cannot be improved. In this last run of GUM, we require $\sum_{j=0}^{n} I(n, 2^{(n-j)})$ Grover iterations. As explained earlier, each Grover iteration requires two calls to a function $f_{\mathbf{x}}$. In addition, we expect to validate $(n+2)$ measured solutions, requiring another $(n+2)$ calls to function $f_{\mathbf{x}}$. Therefore, in the worst case, we expect that RAP requires $\sum_{j=0}^{n} I(n, 2^{(n-j)}) + (n+2) + \sum_{i=0}^{n} \sum_{j=0}^{n} \left(1 - \sum_{j'=0}^{j} P(n, 2^{(n-i)}, 2^{(n-j')})\right) P(n, 2^{(n-i)}, 2^{(n-j)}) I(n, 2^{(n-j)})$ calls to function $f_{\mathbf{x}}$. Figure 4 illustrates the worst-case number of expected calls required by RAP divided by $\sqrt{2^n}$ for $n : 1 \leq n \leq 60$. We observe that the worst-case number of expected calls converges to $13.72\sqrt{2^n}$, implying that: (1) RAP has worst-case expected runtime $O(\eta\sqrt{2^n})$ for solving any discrete optimization problem that has n binary decision variables, and (2) RAP can solve the binary knapsack problem using $O(n\sqrt{2^n})$ operations (in the worst case; thereby matching the performance of the best classical algorithms).

To verify the effective runtime of RAP, we simulate its execution 1000 times for each of the aforementioned 66000 problem instances. The results are summarized in Figure 5 that shows the expected number of operations (divided by $n\sqrt{2^n}$) when solving the binary knapsack problems using HBB and RAP (top left and right panels, respectively). We observe that RAP easily outperforms HBB, and that its performance remains well below its worst-case performance (indicated by the red line in the right top panel). Figure 5 also shows the expected number of operations (divided by $n\sqrt{2^n}$) that are required to find an optimal solution for the first time. Here also, we see that RAP (right bottom panel) outperforms HBB (left bottom panel). Before we conclude, however, that RAP dominates HBB, we first investigate whether the performance of both procedures can be improved.

Figure 3. Illustration of the dynamics of RAP when solving the example binary knapsack problem.

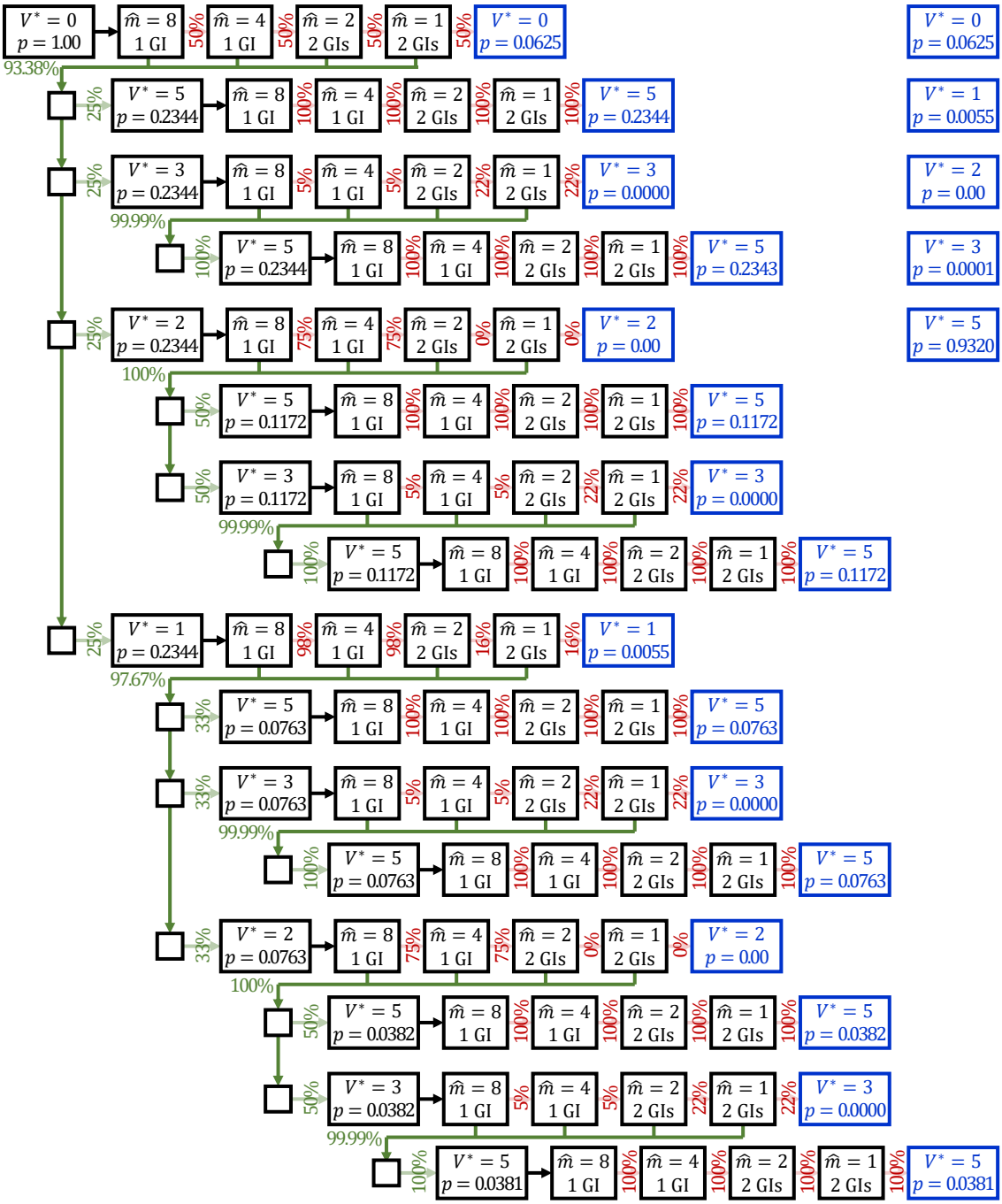


Figure 4. Worst-case expected number of operations divided by $n\sqrt{2^n}$ required by procedure RAP for each value of n : $1 \leq n \leq 60$.

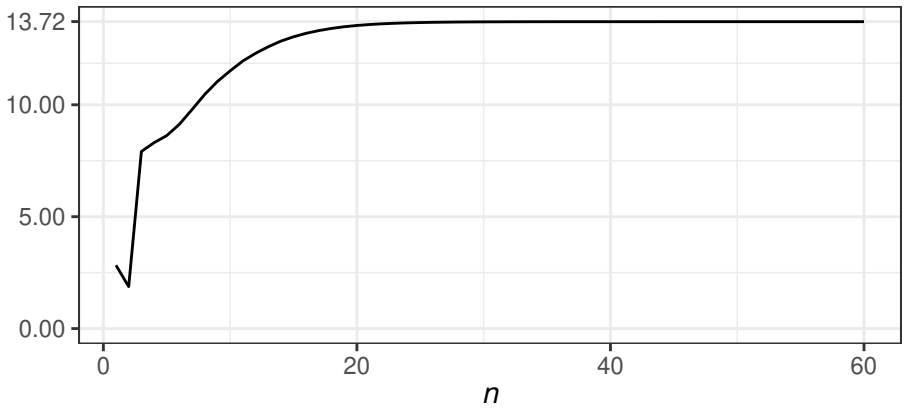
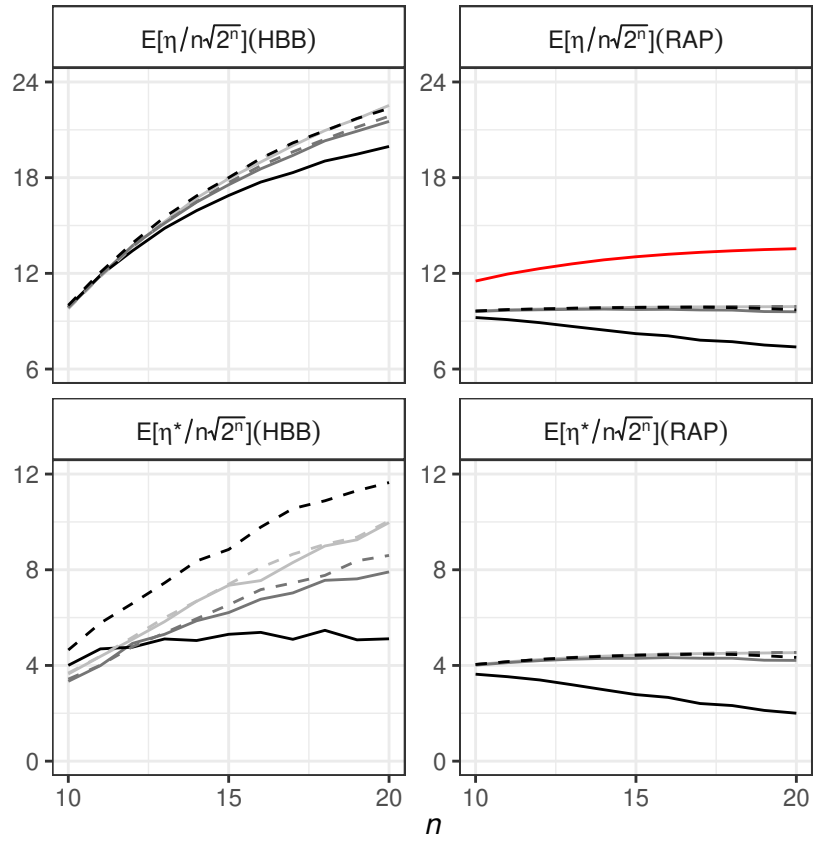


Figure 5. Expected number of operations divided by $n\sqrt{2^n}$ required to solve a binary knapsack problem using HBB and RAP, as well as expected number of operations divided by $n\sqrt{2^n}$ required to find an optimal solution for the first time using both procedures, when solving 1000 binary knapsack problems for each problem set and for each value of n : $10 \leq n \leq 20$ (lines of sets 1 and 4 are light gray, sets 2 and 5 are dark gray, sets 3 and 6 are black, sets 1, 2, and 3 are full, and sets 4, 5, 6 are dashed). The worst-case expected performance of procedure RAP is indicated by the red line (see also Figure 4).



4. Improving procedures HBB and RAP

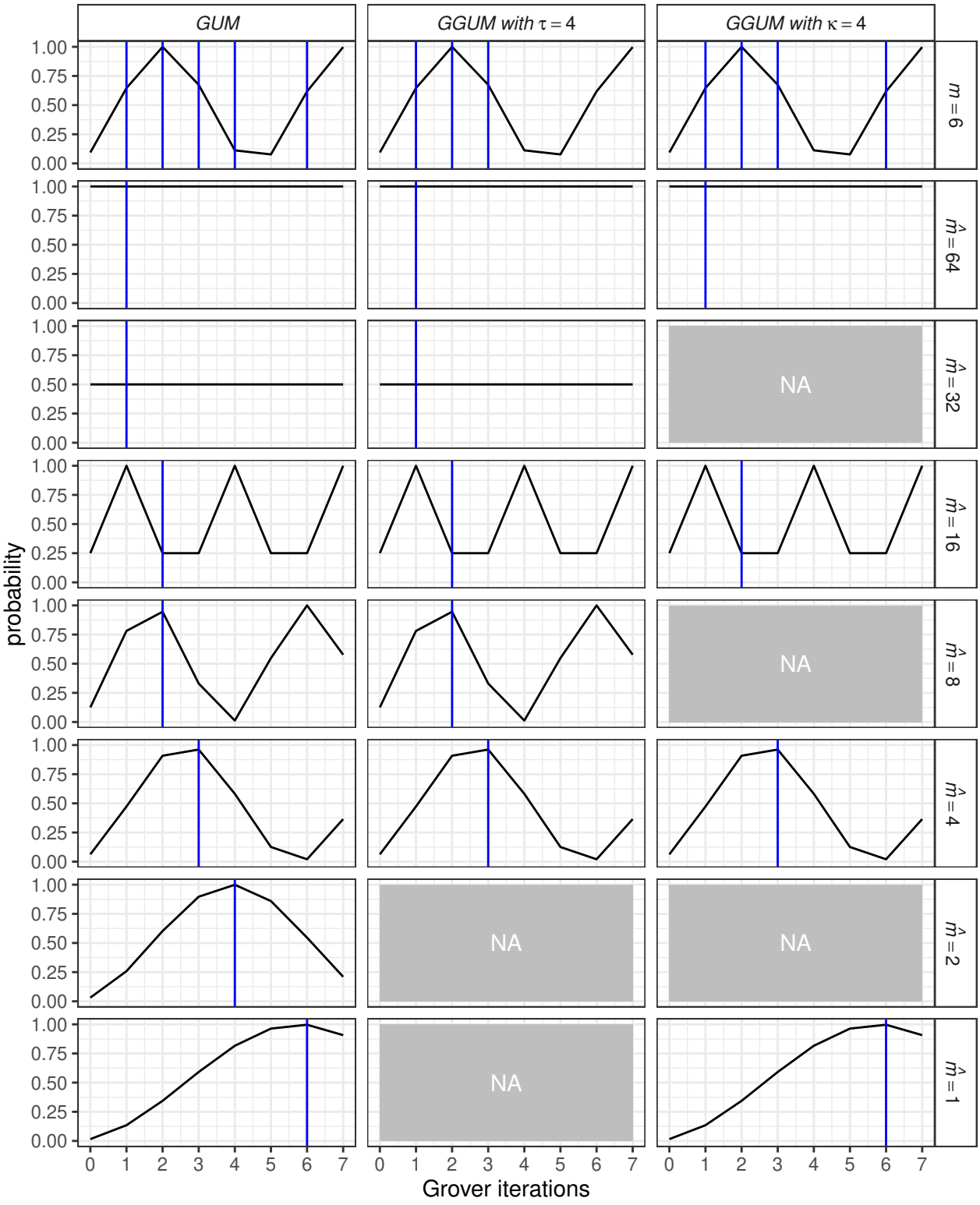
Both HBB and RAP can be improved in several ways. First, a heuristic solution procedure can be used to provide a better initial solution (resulting in less recursions/iterations to be performed by both procedures, respectively). Second, GUM lies at the core of the performance of HBB and RAP. Therefore, if we can improve GUM, we also improve HBB and RAP. Note that, as the true number of valid solutions (i.e., m) is unknown, GUM takes a sample of assumed numbers of valid solutions (i.e., $\widehat{m}$) by initializing $\widehat{m}$ as the total number of solutions, and by halving $\widehat{m}$ in subsequent iterations. If no solution is found after evaluating $\widehat{m} = 1$, the procedure stops. Better sampling methods, however, may exist (e.g., alternatives are discussed by Boyer et al. (1996) and Aaronson (2018)). In this section, we investigate alternative sampling methods, and generalize GUM by: (1) dividing $\widehat{m}$ by κ rather than by 2, and (2) stopping the procedure once $\widehat{m} \leq \tau$. Figure 6 illustrates the sample of values of $\widehat{m}$ (and the corresponding number of Grover iterations) that is used in a system that has 6 qubits and 6 valid solutions: (1) in case of regular GUM (left column), (2) if we let $\tau = 4$ (middle column), and (3) if we let $\kappa = 4$ (right column). For instance, if $\tau = 4$, we do not evaluate values $\widehat{m} = 2$ and $\widehat{m} = 1$. If, on the other hand, $\kappa = 4$, we skip almost half of the values of $\widehat{m}$. If less values of $\widehat{m}$ are evaluated, less Grover iterations are required, however, it also becomes less likely that we measure a valid solution (in the three cases, after evaluating all values of $\widehat{m}$, we have a probability of 1.00, 1.00, and 0.99 to measure a valid solution, respectively). In addition, as a another improvement, note that, currently, HBB and RAP do not use any information of past runs of GUM. If a solution was found assuming a particular value of $\widehat{m}$, however, it makes sense to assume that in a subsequent (more restricted) search there are at most $\widehat{m}$ valid solutions (i.e., it does not make sense to once more reinitialize $\widehat{m}$ as the total number of solutions; we can use the value of $\widehat{m}$ for which a solution was found in a previous search to start searching in a subsequent, more restricted search). These ideas are reflected in Algorithm 5 (referred to as GGUM in what follows) that generalizes GUM.

Algorithm 5: Generalized procedure that uses Grover's algorithm to find a valid solution when the number of valid solutions is unknown in a system that has n binary decision variables, and where $f_{\mathbf{x}}$ returns 1 if solution $\mathbf{x}$ is valid (and 0 otherwise).

```
procedure GGUM(n, m̂)
    do
        perform I(n, m̂) iterations of Grover's algorithm equipped with f_x;
        measure basis state |x⟩;
        if f_x = 1 then
            return x and m̂;
        else if m̂ > τ then
            m̂ = m̂/κ;
        else
            return ∅;
    while true;
```

Next to these (potential) improvements, HBB may also benefit from classical techniques to improve the performance of a branch-and-bound procedure. For instance, when solving the binary knapsack problem,

Figure 6. Sample of values of $\hat{m}$ (and corresponding number of Grover iterations) that are used in case of regular `GUM`, `GGUM` with $\tau = 4$, and `GGUM` with $\kappa = 4$. Each graph presents the probability to measure a valid solution in a system that has $n = 6$ qubits if $m = 6$ (top row), $m = 64$ (second row), $m = 32$ (third row), $m = 16$ (fourth row), $m = 8$ (fifth row), $m = 4$ (sixth row), $m = 2$ (seventh row), and $m = 1$ (eight row). In rows 2 - 8, the blue line represents the number of Grover iterations given by $I(n, \hat{m})$. In the top row, the blue lines correspond to the sample of values of $\hat{m}$ used by each approach to measure any of the $m = 6$ valid solutions.



items can be sorted by their ratio of value over weight, and Algorithm 6 can be used to calculate an upper bound on the maximum value that can still be obtained in any node at level i (hereafter referred to as `UBi`; see also Dantzig (1957)). If this upper bound does not exceed the best-found solution, we can fathom the node. Note that, in contrast to `HBB`, `RAP` cannot not benefit from this upper bound procedure.

Algorithm 6: Procedure that calculates an upper bound on the value of the current knapsack solution (that has value V and remaining weight capacity R) in a node at level i of the search tree. Note that we assume that items are sorted by their ratio of value over weight.

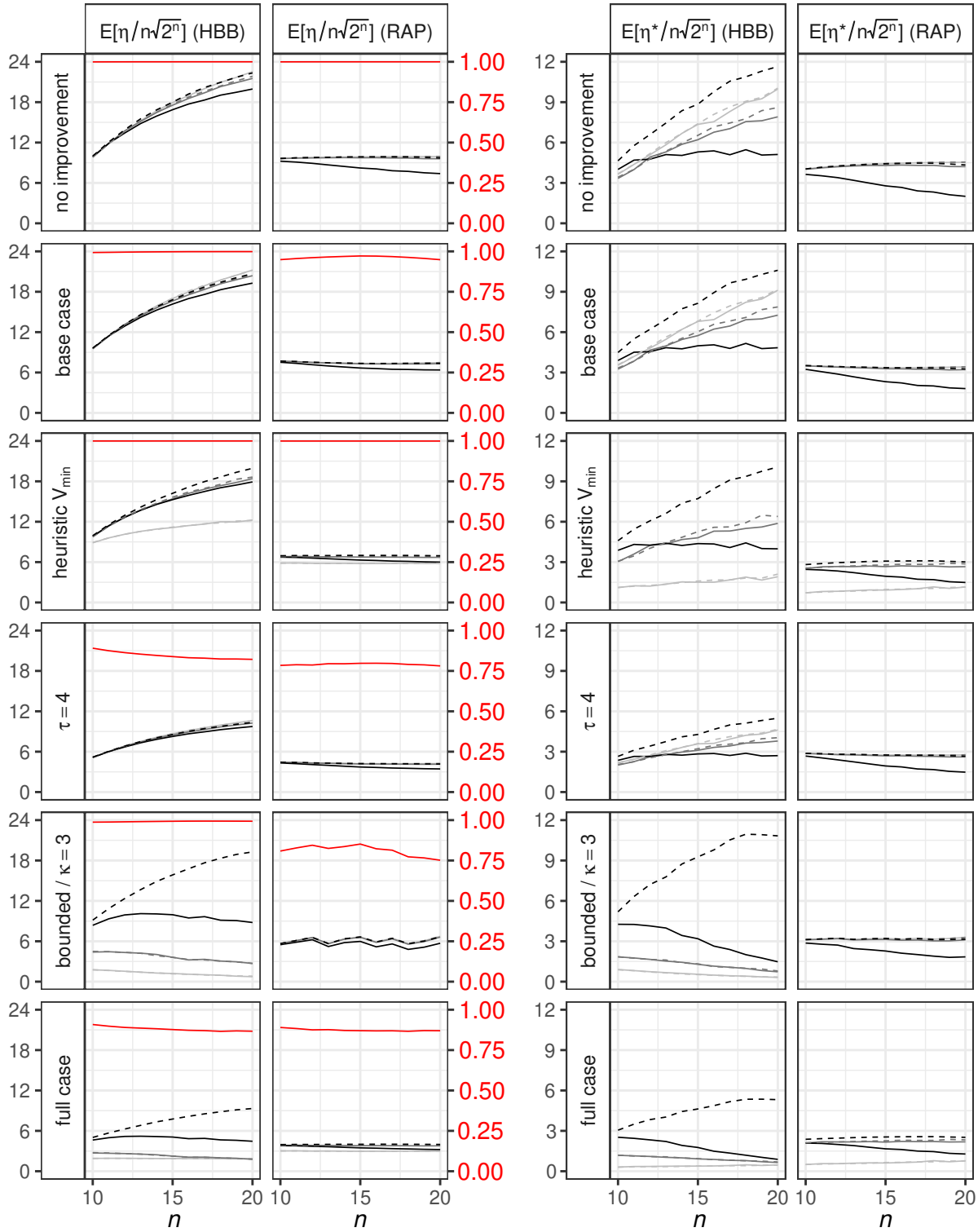
```
procedure UBi(V, R, i)
    for j = i to n do
        if w_j ≤ R then
            let V = V + v_j and R = R − w_j;
        else
            let V = V + (R / w_j) v_j;
            break;
    return V;
```

In what follows, we perform a computational experiment in which we solve 66000 instances of the binary knapsack problem, and assess the impact of:

- using information of the past run of `GGUM` to determine the initial value of $\widehat{m}$ in the subsequent run of `GGUM` (referred to as the "base case"; all improvements mentioned below depart from this base case).
- using a heuristic procedure to calculate a lower bound that serves as an initial solution. The heuristic procedure first sorts items based on their ratio of value over weight (higher ratios are ranked first). Next, we initialize an empty knapsack, and add items in order of the list. If an item cannot be added (because its weight exceeds the remaining weight capacity), we move to the next item in the list. The procedure stops if all items in the list have been processed.
- assuming a value of $\tau = 4$ (i.e., $\widehat{m} = 1$ and $\widehat{m} = 2$ are not evaluated by `GGUM`).
- assuming a value of $\kappa = 3$ (i.e., we evaluate $\widehat{m} = 2^n$, $\widehat{m} = {}^{2^n}/_3$, $\widehat{m} = {}^{2^n}/_9$ etc.).
- using upper bound `UBi` to fathom nodes when using `HBB`.

The results of the experiment are summarized in Figure 7. For different scenarios, Figure 7 reports the expected number of operations (divided by $n\sqrt{2^n}$) required by `HBB` and `RAP` (left two columns) as well as the expected number of operations (divided by $n\sqrt{2^n}$) required to find an optimal solution for the first time (right two columns). Figure 7 also shows the average probability to find the optimal solution for each of the scenarios/procedures (red line in the left two columns; note that the expected number of operations to find an optimal solution for the first time only is calculated for those instances for which the optimal solution was found). For comparison, we also included the results of `HBB` and `RAP` without improvements (first row; these results are identical to those presented in Figure 5).

Figure 7. Expected operations (divided by $n\sqrt{2^n}$) to solve a binary knapsack problem using `HBB` and `RAP` as well as the expected operations (divided by $n\sqrt{2^n}$) to find an optimal solution for the first time, when solving 1000 knapsack problems for each problem set, for various values of n, and for various scenarios (lines of sets 1 and 4 are light gray, sets 2 and 5 are dark gray, sets 3 and 6 are black, sets 1, 2, and 3 are full, and sets 4, 5, 6 are dashed). A red line shows the average probability to identify the optimal solution.



A first conclusion that can be drawn by observing Figure 7 is that initializing $\widehat{m}$ as the last value of $\widehat{m}$ in the previous run (base case; reported in row 2) has a significant impact on the number of required operations without significantly reducing the probability to measure the optimal solution. Next, even though the use of a lower bound (row 3) has a positive impact, the effect is limited except for sets 1 and 4 (where items have uncorrelated values and weights). This can be explained by the fact that items in sets 2 and 5 have similar ratios (values and weights are correlated) and items in sets 3 and 6 have ratios close to 1 (item i has value $v_i = 3\lceil w_i/3 \rceil$)). As a result, for these sets, our lower bound (based on the ratio of value over weight) may not be very effective. In row 4, on the other hand, we see that letting $\tau = 4$ once more has a significant effect on the number of required iterations without impacting the probability to measure the optimal solution too much. In contrast, in row 5, we evaluate the impact of letting $\kappa = 3$ for `RAP` (second and fourth column). Even though we are able to reduce the number of required operations, we also observe that the probability to measure the optimal solution reduces as n increases. In other words, with $\kappa = 3$, the sample of values of $\widehat{m}$ becomes too sparse to still guarantee that `GGUM` can still measure a valid (and hence optimal) solution. In row 5, we also illustrate the impact of upper bound `UBi` on the performance of `HBB`. We observe a significant reduction of the number of operations required to find the optimal solution (first column) and the number of operations required to find the optimal solution for the first time (third column). The impact is most outspoken for instances where the structure of the problem can be exploited (i.e., instances with uncorrelated and correlated values and weights). For sets 3 and 6, the impact is less clear. For instances in set 6, the effectiveness of `UBi` is limited by the fact that items have a ratio of value over weight that is close to 1. For set 3, this effect is offset by the fact that instances in set 3 have many unique feasible solutions (see also Table 2). Last but not least, in row 6 we evaluate the impact of all improvements (except $\kappa = 3$ and upper bound `UBi` in the case of `RAP`; referred to as "full case" in Figure 7) on the performance of `HBB` and `RAP`. Here we observe that both procedures have roughly the same probability to find the optimal solution, however, `RAP` no longer dominates `HBB`. Only when solving instances of set 6, `RAP` still has the upper hand. Perhaps more importantly, row 6 also illustrates that `HBB` and `RAP` may require (far) less than $O(n\sqrt{2^n})$ operations to find an optimal solution for the first time. This indicates that they may be used as efficient heuristics to obtain (near-) optimal solutions.

5. Combining procedures `HBB` and `RAP`

In this section, we combine both `HBB` and `RAP`. Whereas `HBB` has the disadvantage that a significant amount of time is spent on proving that a (partial) solution is optimal, it has the advantage that it can exploit the structure of a problem (resulting in the removal of subsets of solutions from the search space). `RAP`, on the other hand, minimizes the time spent on proving that a solution is optimal, however, it's disadvantage is that it always searches the entire solution space. To combine the strenghts of both procedures, `HBB` can be equipped with `RAP` at deeper levels of the search tree. This way: (1) we benefit from the speedup created by the branch-and-bound procedure at higher levels of the search tree, (2) we spend less time to prove that a partial solution is optimal in deeper levels of the tree, and (3) we also alleviate the disadvantage of `RAP` because it is used to find a partial solution in a subset of the solution space. In addition, note that at higher

levels of the search tree, there is not so much difference between HBB and RAP. As a result, it makes sense to use RAP mainly at deeper levels of the search tree.

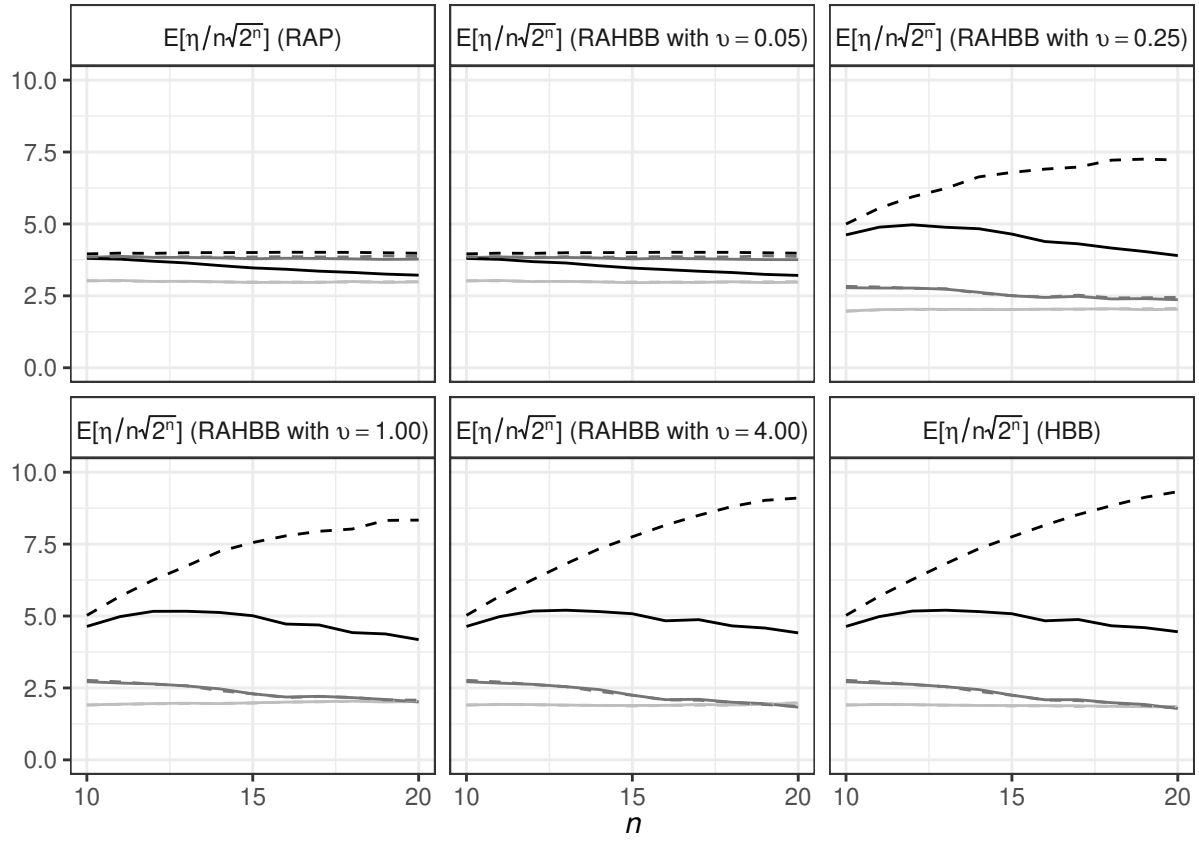
Algorithm 7 (hereafter referred to as RAHBB) presents one possible implementation that uses RAP if the number of Grover iterations required by HBB exceeds a given threshold $n\upsilon$. Procedure RAHBB also uses upper bound UBi to fathom nodes, and is equipped with GGUM rather than with GUM.

Algorithm 7: Hybrid branch-and-bound procedure that uses a random-ascent procedure if the number of required Grover iterations exceeds υn. Both procedures use GGUM to measure a valid solution.

initialize $\mathbf{x}^*$ and V^* using lower bound heuristic and start procedure RAHBB$(\emptyset, 0, W, 2^n, 1)$;
Procedure RAHBB$(\mathbf{x}, V, R, \widehat{m}, i)$
  if UBi$(V, R, i) > V^*$ **then**
    if $n - i < \omega$ **then**
      use classical algorithm to find $\mathbf{x}^\star_{in} \in \mathcal{X}_{in}$ that has highest value $V^\star_{in} > V^* - V$ and $W^\star_{in} \leq R$;
      if $\mathbf{x}^\star_{in} \neq \emptyset$ **then**
        update best-found solution $\mathbf{x}^* = \mathbf{x} + \mathbf{x}^\star_{in}$ and its value $V^* = V + V^\star_{in}$;
    else if $I(n - i, \widehat{m}) \leq \upsilon n$ **then**
      if $w_i \leq R$ **then**
        update current solution $\mathbf{x}' = \mathbf{x} + \{1\}$, its value $V' = V + v_i$, and its remaining weight $R' = R - w_i$;
        Use GGUM$(n - i, \widehat{m})$ to obtain $\widehat{m}'$ and to measure $\mathbf{x}_{(i+1)n} \in \mathcal{X}_{(i+1)n}$ that has value $V_{(i+1)n} > V^* - V'$ and $W_{(i+1)n} \leq R'$;
        if $\mathbf{x}_{(i+1)n} \neq \emptyset$ **then**
          update best-found solution $\mathbf{x}^* = \mathbf{x}' + \mathbf{x}_{(i+1)n}$ and its value $V^* = V' + V_{(i+1)n}$;
          RAHBB$(\mathbf{x}', V', R', \widehat{m}', i + 1)$;
      RAHBB$(\mathbf{x}, V, R, \widehat{m}, i + 1)$;
    else
      do
        Use GGUM$(n - i, \widehat{m})$ to obtain $\widehat{m}$ and to measure $\mathbf{x}'$ that has value $V' > V^*$ and $W' \leq R$;
        if $\mathbf{x}' \neq \emptyset$ **then**
          update best-found solution $\mathbf{x}^* = \mathbf{x} + \mathbf{x}'$ and its value $V^* = V + V'$;
      while $\mathbf{x} \neq \emptyset$;

We use RAHBB to solve the 66000 instances of the binary knapsack problem assuming the same settings as those used in the "full case" discussed in the previous section (i.e., we use a lower bound heuristic to set an initial solution, we let $\tau = 4$, we use upper bound procedure UBi to phatom nodes, and we use the number of valid solutions of the last search as the initial number of valid solutions in the subsequent search). The results are summarized in Figure 8. Figure 8 presents the number of operations (divided by $n\sqrt{2^n}$) that are required by RAHBB for various values of υ. Figure 8 also reports the performance of RAP (top left panel) and HBB (bottom right panel). We can observe that, for sufficiently small υ, the performance of RAHBB is almost identical to that of RAP (i.e., RAHBB skips HBB, and immediately performs RAP). On the other hand, if υ is

Figure 8. Expected number of operations divided by $n\sqrt{2^n}$ required to solve a binary knapsack problem using HBB, RAP, and RAHBB for various values of υ, when solving 1000 binary knapsack problems for each problem set, for each value of $n : 10 \leq n \leq 20$ (lines of sets 1 and 4 are light gray, sets 2 and 5 are dark gray, sets 3 and 6 are black, sets 1, 2, and 3 are full, and sets 4, 5, 6 are dashed).



sufficiently large, RAHBB is identical to HBB (i.e., RAHBB never performs RAP). Figure 8 shows that, depending on our ability to exploit the structure of the problem, a combination of HBB and RAP may be used to reduce the overall expected number of operations.

6. Generalizing procedures HBB and RAP

Until now, discrete optimization problems with binary decision variables have been considered. Procedures HBB and RAP (and by extension RAHBB), however, can easily be adapted to solve discrete optimization problems that consider discrete decision variables. If we assume that a decision variable can take on 2^b values, we require b qubits to represent that decision variable (i.e., in the case of binary decision variables, $b = 1$, and a single qubit suffices to represent a decision variable). If each of the n decision variables requires b qubits, GGUM is applied to nb qubits rather than to n qubits. As a result, there are 2^{nb} possible solutions, and GGUM will evaluate up to $log_2(2^{nb}) = nb$ values of $\widehat{m}$. For each value of $\widehat{m}$, GGUM performs $I(2^{nb}, \widehat{m})$ Grover iterations, and each Grover iteration requires two calls to a function that evaluates a solution of the discrete optimization problem. If we assume it takes η operations to evaluate a solution $\mathbf{x}$, HBB and RAP (or a combination thereof)

require $O(\eta\sqrt{2^{nb}})$ operations.

7. Conclusion

In Part I of this paper, we present a binary search procedure (BSP) that uses Grover's algorithm (implemented as GUM) to effectively solve a discrete optimization procedure that has n decision variables using $O(L\eta\sqrt{2^n})$ operations, where: (1) L is a logarithmic function of the difference between the initial bounds on the optimal solution value, and (2) η denotes the number of operations required to evaluate a solution. In this paper, we improve this result. For this purpose, we propose two new classical algorithms that use GUM to measure a solution: a hybrid branch-and-bound procedure (HBB) and a random-ascent procedure (RAP). We use both procedures to solve 66000 instances of the binary knapsack problem, and show that they require $O(\eta\sqrt{2^n})$ operations to solve a binary knapsack problem that has n items (i.e., they match the performance of the best classical algorithms). The initial results show that RAP dominates HBB. In addition, we show that, even in the worst-case, RAP expects to use $O(\eta\sqrt{2^n})$ operations when solving a discrete optimization problem that has n binary decision variables.

To further improve the performance of HBB and RAP, we assess the impact of several improvements when solving the binary knapsack problem. Because both procedures rely on GUM to find valid solutions, we first generalize GUM in the following way: (1) if a valid solution was found assuming $\widehat{m}$ solutions in the last search, we assume $\widehat{m}$ is the initial number of valid solutions in the next search (that considers a more restricted problem that has at most $\widehat{m}$ valid solutions), (2) in each iteration, rather than letting $\widehat{m} = \widehat{m}/2$, we let $\widehat{m} = \widehat{m}/\kappa$, and (3) we stop the procedure after processing $\widehat{m} = \tau$ rather than after $\widehat{m} = 1$. The generalized procedure (GGUM) allows us to evaluate different strategies in which different samples of $\widehat{m}$ are used to find a valid solution (note, however, that further research is required to develop a procedure that samples $\widehat{m}$ such that the expected number of Grover iterations is minimized while maximizing the probability to find a valid solution). Next, we also investigate the impact of using a heuristic to initialize the starting solution (rather than assuming an initial solution $V^* = 0$). To investigate the potential of techniques that improve the performance of the branch-and-bound of HBB, we also adopt an upper bound procedure to fathom nodes (note that this upper bound procedure cannot be used by RAP). The results of the experiment show that, even though we significantly improve performance, both procedures still require $O(\eta\sqrt{2^n})$ operations to solve a binary knapsack problem that has n items. The results, however, also show that HBB can outperform RAP if the structure of the problem can be exploited efficiently. In addition, the results indicate that HBB and RAP can be used as heuristics that are able to find (near-) optimal solutions using (far) less than $O(\eta\sqrt{2^n})$ operations.

Next, to improve HBB and RAP, we also generalize both procedures, and argue that they can be used to solve any discrete optimization problem using $O(\eta\sqrt{2^{nb}})$ operations, where b is the number of discrete values that can be assigned to any of the n decision variables. As such, procedures HBB and RAP may be seen as general-purpose solvers that can solve well-known combinatorial problems as well as complex non-linear integer programming problems for which no dedicated algorithms exist. This is where the power of both procedures lies: their general applicability to solve any discrete optimization problem. HBB and RAP not only offer state-of-the-art performance to solve discrete optimization problems, they can also be used as heuristics

that can find (near-)optimal solutions using (far) less than $O(\eta\sqrt{2^{nb}})$ operations. In addition, as a branch-and-bound procedure, HBB allows to exploit the structure of a problem to further improve performance. In contrast, RAP can be used to solve problems that have no clear structure or that have a structure that is difficult to exploit. To combine the strengths of HBB and RAP, we propose another procedure (referred to as RAHBB) whose performance can be tailored depending on how effectively we are able to exploit the structure of the problem.

Note that procedures HBB and RAP (or a combination thereof) can easily run in parallel. In the case of RAP, for instance, given i quantum processors, i instances of RAP can be running at the same time, and we expect an optimality gap of $1/(i+1)$ after completing a first iteration of RAP on all processors (e.g., if we have 20 quantum processors, we expect an optimality gap of less than 5 percent). In addition, we don't have to wait until completing an iteration of RAP to already share best-found solutions. In fact, for every value of $\widehat{m}$ that is evaluated using GGUM, if a solution is found that improves the best-found solution, subsequent searches on all quantum processors can use this best solution as a lower bound. It would be interesting to investigate the computational gains that can be achieved this way.

Our work has several limitations. First, future research should evaluate the potential of HBB and RAP to solve discrete optimization problems other than the binary knapsack problem. By comparing the performance of HBB and RAP with that of state-of-the-art classical algorithms, problems may be identified that are more suited to be solved by quantum algorithms. Second, procedures HBB and RAP can be extended such that they can also solve more general optimization problems (e.g., mixed-integer programming problems). Third, we focus on Grover-based algorithms. Using such algorithms, it may be difficult to solve discrete optimization problems using less than $\sqrt{2^n}$ Grover iterations (and hence calls to a function $f_{\mathbf{x}}$). This is also illustrated in Figure 9 that shows both $\pi 4^{-1}\sqrt{2^n}$ (i.e., the number of Grover iterations required to maximize the probability to measure a valid solution using Grover's algorithm) as well as n^2 (i.e., the number of operations required for a procedure that has quadratic complexity). It is clear that, as n increases, an optimization procedure that relies on Grover's algorithm will always require at least $O(\eta\sqrt{2^n})$ operations. Amplitude amplification is a generalization of Grover's algorithm that aims to solve exactly this problem (see e.g., Brassard et al., (2002)). Whereas Grover's algorithm starts with a "uniform superposition" of qubits (i.e., a superposition where each solution has equal probability), amplitude amplification uses a superposition that allows to measure the target state using (far) less iterations. This is illustrated in Figure 10 that illustrates the uniform superposition (used by Grover's algorithm), a "good" superposition, and a "bad" superposition. A good superposition uses information of the optimization problem to reduce the number of iterations required to measure a valid solution. As shown by Creemers and Pérez (2023), however, finding such a good superposition may not be that easy. In fact, Creemers and Pérez show that, following logical rules, it is possible to end up with a bad superposition that performs worse than a uniform superposition. Nevertheless, more research is required to determine the potential of amplitude amplification when solving discrete optimization problems.

Next to Grover-based algorithms, adiabatic quantum computation may also be of interest. Adiabatic quantum computing exploits the fact that in physics everything tends to seek a minimum energy state. Using adiabatic quantum computing, discrete optimization problems can be formulated as energy minimization problems. Starting from an initial state, the adiabatic theorem is then used to evolve the quantum system

Figure 9. Number of Grover iterations required to maximize the probability to measure a valid solution in a system that has one valid solution for various values of n. The area in blue is the area that can be sampled if we use up to n^2 Grover iterations.

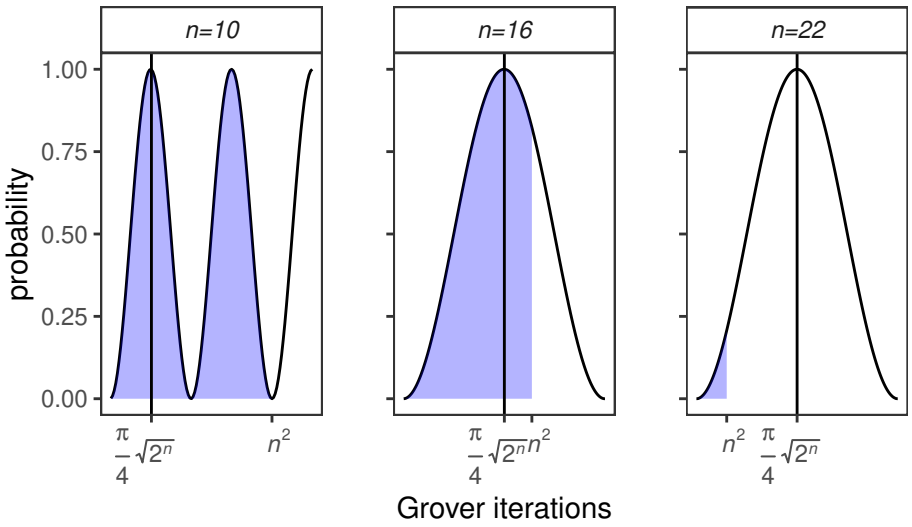
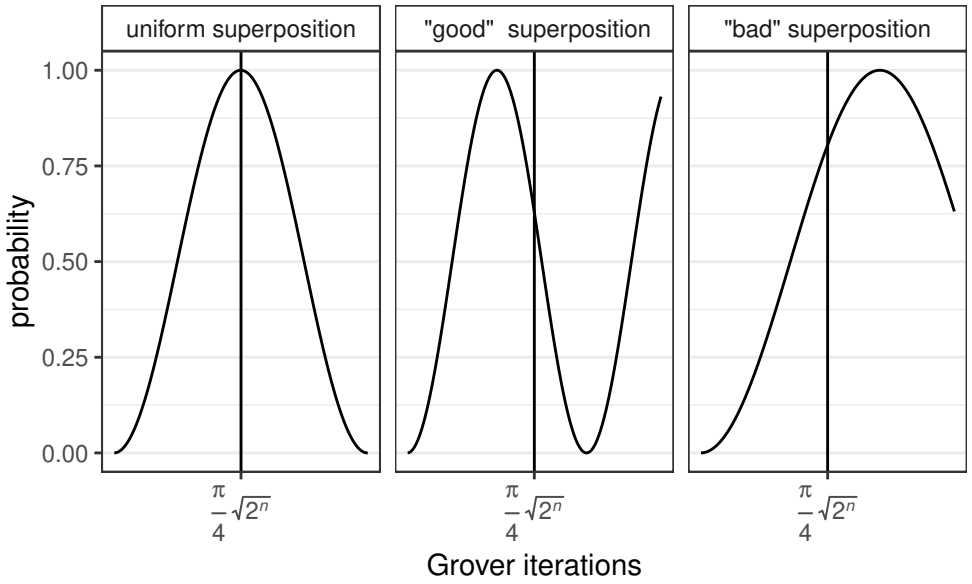


Figure 10. Comparison of a uniform superposition (used by Grover's algorithm), a "good" superposition, and a "bad" superposition when measuring a valid solution in a system that has 1 valid solution and n qubits.



towards a low-energy state. This low-energy state corresponds to the optimal or near-optimal solution of the problem. Popular algorithms that rely on the adiabatic theorem (Born and Fock, 1928) include the Quantum Annealer algorithm (QA; see Apolloni et al. (1989)), the Quantum Approximate Optimization Algorithm (QAOA; see Farhi et al., (2014)), and the Variational Quantum Eigen-Solver algorithm (VQE; see Peruzzo et al. (2014)).

As already mentioned in Part I of this paper, we assume the existence of a noiseless universal quantum computer that can effectively run HBB and RAP. If we look at IBM's (ambitious) roadmap to develop such a computer, it becomes clear that it may take many more years before it will hit the market. Once a noiseless quantum computer does become available, however, state-of-the-art procedures such as HBB and RAP (or a combination thereof) can serve as general-purpose solvers that can compete with e.g., commercial off-the-shelf solvers such as CPLEX and GUROBI, and the field of discrete optimization may be reshaped forever. Even though the quantum revolution is not there yet, it is at our doorstep.

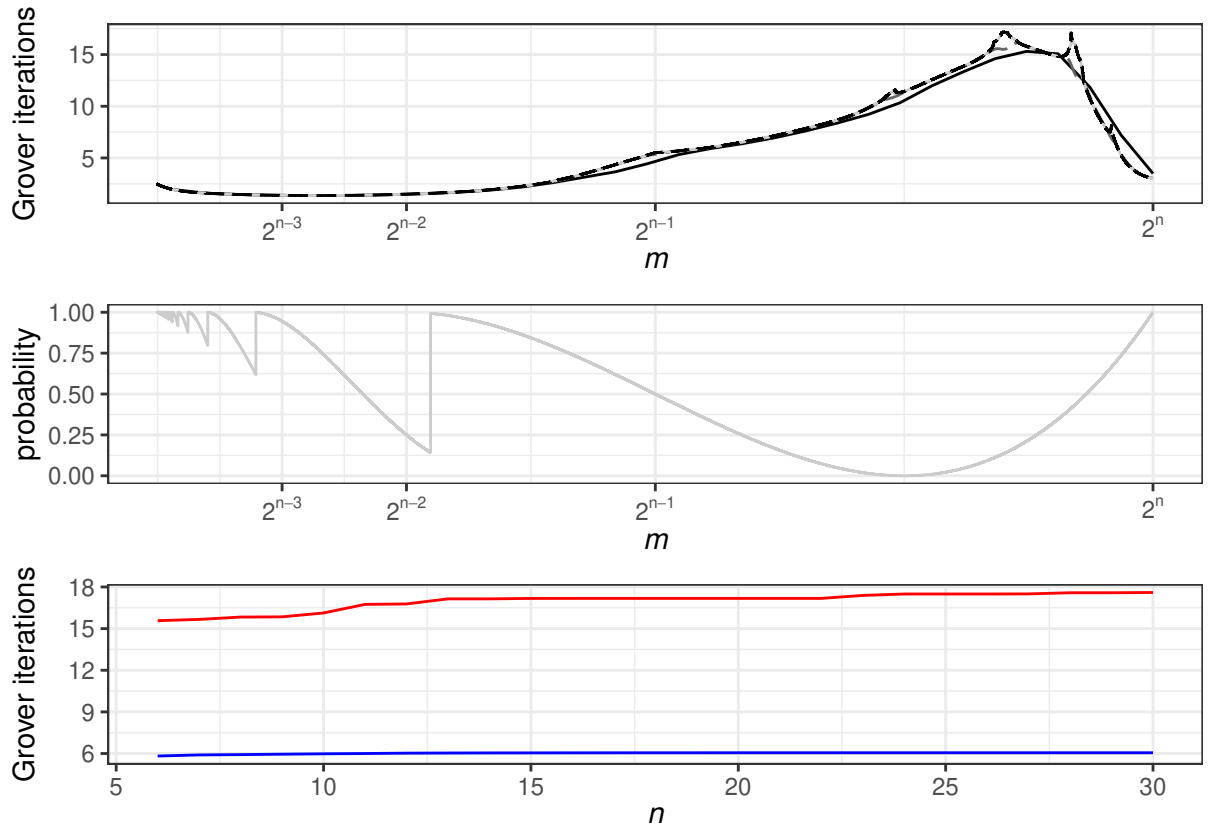
The contribution of Part II of this paper may be summarized as follows: (1) we propose two new procedures that can solve any discrete optimization problem using $O(\eta\sqrt{2^{nb}})$ operations, (2) we use these new procedures to solve the binary knapsack problem, and show that they match the performance of the best classical algorithms, (3) we introduce several improvements that further increase the performance of both procedures, (4) both procedures can also be used as heuristics to find (near-)optimal solutions using (far) less than $O(\eta\sqrt{2^{nb}})$ operations, (5) we present another procedure that combines the strengths of the two aforementioned procedures, and (6) we conclude that these procedures are the new state-of-the-art in quantum computing that one day can be used as general-purpose solvers that can solve any discrete optimization problem.

Appendix

Performance of GUM

In this section, we perform a computational experiment to investigate the performance of GUM when measuring any of the m valid solutions in a system that has n qubits. The results of the experiment are summarized in Figure 11. The top panel of Figure 11 shows the number of operations (divided by $\sqrt{m^{-1}2^n}$) required by GUM for various values of m and n (note that we divide by $\sqrt{m^{-1}2^n}$ to show that GUM requires $O(c\sqrt{m^{-1}2^n})$ operations, where c is a constant). We observe that the expected number of operations is rather constant if $m < 2^{(n-2)}$. Only for values of $m > 2^{(n-2)}$, the expected number of operations starts to increase. This is not surprising as approximation $I(n,m)$ (that is used by GUM to determine the number of Grover iterations) is often inaccurate if $m > 2^{(n-2)}$ (see also the middle panel of Figure 11 that shows the probability that Grover's algorithm is able to measure any of the m solutions if $I(n,m)$ Grover iterations are used). To resolve this problem, we can evaluate e.g., 10 random solutions before running GUM. If we have a probability of 50 percent to find a valid solution, with 10 trials we are at least 99.90 percent certain to find a valid solution. If no valid solution was found after 10 trials, we can safely assume there are (far) less than $2^{(n-2)}$ valid solutions, and we run GUM. Last but not least, the bottom panel of Figure 11 shows the average and the maximum number of expected operations (divided by $\sqrt{m^{-1}2^n}$) required by GUM over all values of m, for various values of n. We

Figure 11. The top panel shows the expected number of operations divided by $\sqrt{m^{-1}2^n}$ required by GUM for various values of m, and for $n = 6$ (black full line), $n = 10$ (dark gray dashed line), $n = 16$ (black dashed line), and $n = 22$ (gray full line). The middle panel shows the probability to measure one of the m valid solutions using Grover's algorithm if $I(n, m)$ Grover iterations are used. The bottom panel shows the average (blue) and maximum (red) number of expected operations divided by $\sqrt{m^{-1}2^n}$ required by GUM over all values of m, for various values of n.



observe that the average and the maximum number of operations converge as n increases. In addition, the expected number of operations (divided by $\sqrt{m^{-1}2^n}$) shown in the top panel also converges as n increases. As a result, we conclude that GUM requires $O(\sqrt{m^{-1}2^n})$ operations to find any of the m valid solutions in a set of 2^n solutions.

Assessing the performance of a quantum algorithm

To assess the performance of a quantum algorithm, we essentially have three options: (1) to calculate all possible outcomes and their probabilities, (2) to use an existing quantum simulator, or (3) to simulate the performance yourself. For evaluating the performance of BSP, we use the first 2 options, and verify the procedure in Qiskit (IBM's software development kit that allows to simulate or operate a universal quantum gate computer) and calculate all possible outcomes and their probabilities. This allows us to assess (exactly) the expected runtime as well as the probability to measure the optimal solution. Even though Qiskit is a powerful tool, it is not fit for performing large computational experiments. In addition, calculating all possible

outcomes and their probabilities is often intractable. That is why we decided to simulate the performance of HBB and RAP. In such a simulation, (common) random numbers are used to determine whether or not Grover's algorithm is able to measure the target basis state. For instance, in a system that has $n = 3$ qubits and $m = 1$ target basis state, we can use simulation to assess the performance of GUM. In a first step, GUM assumes $\widehat{m} = 2^n = 8$, and performs $I(3, 8) = 1$ Grover iteration. With 1 Grover iteration, we have a 78.13 percent probability to measure the target basis state. As such, in our simulation, we draw a random number in $[0, 1]$. If this random number is smaller than (or equal to) 0.7813, we conclude that Grover's algorithm has identified the target basis state after 1 Grover iteration. If, on the other hand, the random number is larger than 0.7813, GUM continues, and lets $\widehat{m} = \widehat{m}/2 = 4$. Once more, we perform $I(3, 4) = 1$ Grover iteration. Again, we draw a random number, and have 78.13 percent chance to identify the target basis state. If, twice in a row, we draw a random number larger than 0.7813, we continue with a third step in which GUM lets $\widehat{m} = \widehat{m}/2 = 2$. In this case, $I(3, 2) = 2$, and we have a 94.53 percent probability to identify the target basis state after 2 Grover iterations. Again, we draw a random number, and only if it is larger than 0.9453, we perform a last step where $\widehat{m} = \widehat{m}/2 = 1$. In this last step, $I(3, 1) = 2$, and we are once more 94.53 percent certain to identify the target basis state. After drawing another random number we either identify the target basis state or conclude that no valid solution could be found (note that this requires us to draw two random numbers larger than 0.7813 and two random numbers larger than 0.9453). If simulated sufficient times, the performance of GUM (and hence also the performance of HBB and RAP that rely on GUM) can be approximated with sufficient confidence.

Better approximating the number of Grover iterations

As we have shown in Part I of this paper, the ability of Grover's algorithm to measure a target entry in a database that has 2^n entries depends heavily on the number of Grover iterations that is used. The number of Grover iterations is usually determined by means of approximation $I(n, m) = \lceil \pi 4^{-1} \sqrt{m^{-1} 2^n} \rfloor$. This approximation, however, does not always result in a high probability to measure a target entry, as is also shown in Figure 12 that compares the performance of approximation $I(n, m) = \lceil \pi 4^{-1} \sqrt{m^{-1} 2^n} \rfloor$ with that of $\max(1, \lfloor \pi 4^{-1} \sqrt{m^{-1} 2^n} \rfloor)$. If the real number of target entries is unknown, a sampling method (that evaluates several assumed numbers of target entries) should maximize the probability to measure the target entry while minimizing the expected number of required Grover iterations. As argued in Section 4, such a sampling method may benefit the performance of procedures HBB and RAP.

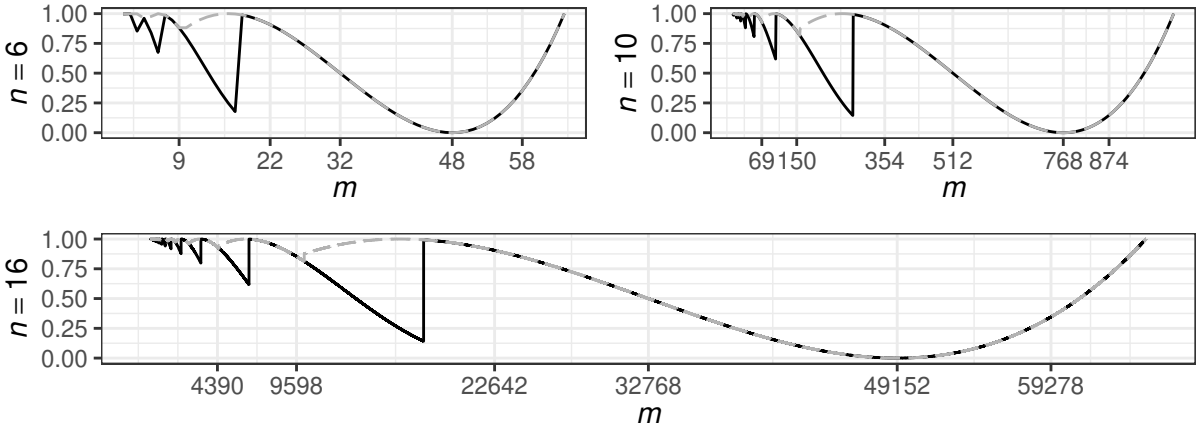
References

Aaronson, S. 2018. *Introduction to Quantum Information Science*. Lecture Notes, UT Austin.

Ambainis, A., Balodis, K., Iraids, J., Kokainis, M., Prūsis, K., and Vihrovs J. 2018. Quantum speedups for exponential-time dynamic programming algorithms, unpublished preprint at https://doi.org/10.48550/arXiv.1807.05209.

Apolloni, B., Carvalho, C., and De Falco, D. 1989. Quantum stochastic optimization. *Stochastic Processes and their Applications*, **33**(2), 233–244.

Figure 12. Comparison of the performance of approximation $\lceil \pi 4^{-1}\sqrt{m^{-1}2^n} \rceil$ (black full line) and approximation $\max(1, \lfloor \pi 4^{-1}\sqrt{m^{-1}2^n} \rfloor)$ (gray dashed line). For each value of $n : n \in \{6, 10, 16\}$, the figure shows the probability to measure a target basis state for various values of m.



Born, M. and Fock, V. 1928. Beweis des adiabatensatzes. *Z. Phys.*, **51**(3–4), 165–180.

Brassard, G., Høyer, P., and Tapp, A. 1998. Quantum counting. In Larsen, K.G., Skyum, S., and Winksel, G. (Eds.), *Automata, Languages and Programming*, 820–831. Springer Berlin Heidelberg, Berlin, Heidelberg.

Brassard, G., Høyer, P., Mosca, M., and Tapp, A. 2002. Quantum amplitude amplification and estimation. In Lomonaco, S.J. and Brandt, H.E. (Eds.), *Quantum computation and information*, 53–74. Amer. Math. Soc., Providence, RI.

Boyer, M., Brassard, G., Høyer, P., and Tapp, A. 1996. *Tight Bounds on Quantum Searching*. Proceedings of the Fourth Workshop on Physics and Computation, New England Complex Systems Institute.

Cerf, N. J., Grover, L. K., and Williams, C. P. 2000. Nested quantum search and structured problems. *Phys. Rev. A*, **61**(3), 1–14.

Creemers, S., Pérez, L. 2023. Discrete optimization: Limitations of existing quantum algorithms, unpublished preprint.

Dantzig, G.B. 1957. Discrete-variable extremum problems. *Operations Research*, **5**(2), 266–288.

Farhi, E., Goldstone, J., and Gutmann, S. 2014. A Quantum Approximate Optimization Algorithm, unpublished preprint at https://doi.org/10.48550/arXiv.1411.4028.

Grinko, D., Gacon, J., Zoufal, C., and Woerner, S. 2021. Iterative quantum amplitude estimation. *npj. Quantum Inf.*, **7**(52).

Grover, L. K. 1996. A fast quantum mechanical algorithm for database search. *Proc. Annu. ACM Symp. Theory Comput.*, 212–219.

Kellerer, H., Pferschy, U., and Pisinger, D. 2004. *Knapsack Problems*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kitaev, A.Y. 1995. Quantum measurements and the Abelian stabilizer problem, unpublished preprint at https://doi.org/10.48550/arXiv.quant-ph/9511026.

Montanaro, A. 2016. Quantum walk speedup of backtracking algorithms, unpublished preprint at https://doi.org/10.48550/arXiv.1509.02374.

McArdle, S., Jones, T., Endo, S., Li, Y., Benjamin, S.C., and Yuan., X. 2019. Variational ansatz-based quantum simulation of imaginary time evolution. *npj Quantum Information*, **5**(1) 75–75.

Nielsen, M., Chuang, I. 2010. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, Cambridge.

Peruzzo, A., McClean, J., Shadbolt, P., Y., M.-H., Z., X.-Q., Love, P.J., Aspuru-Guzik, A., and O'brien, J.L. 2014. A variational eigenvalue solver on a photonic quantum processor *Nature communications*, **5**(1), 4213.

Pisinger, D. 2005. Where are the hard knapsack problems? *Comput. Oper. Res.*, **32**(9), 2271–2284.

Shor, P.W. 1994. Algorithms for quantum computation: Discrete logarithms and factoring. *Proc. Annu. Symp. FOCS*, 124–134.

Chapter 3

Limitations of existing Quantum Discrete Optimization algorithms

The content of this chapter is presented in the form of a paper titled "**Discrete optimization: Limitations of existing quantum algorithms**".

3.1 Chapter 3 introduction

In this thesis chapter, we explore the constraints and challenges associated with several widely recognized quantum algorithms. Our initial focus is on the quantum counting algorithm proposed by Brassard et al. 1998, (hereafter referred to as QCB). The objective of a quantum counting algorithm is to provide an approximation of the number of valid solutions (m) within a set of 2^n solutions. This approximation is particularly crucial in the context of Grover's algorithm, where knowledge of m enables the determination of the optimal number of iterations for maximizing the probability of measuring any of the m valid solutions. However, our investigation reveals that QCB necessitates 2^n Grover iterations for an accurate estimation of m, equivalent to the performance of a brute-force classical approach. Furthermore, we establish that any quantum counting algorithm, not limited to QCB, is overshadowed by GUM (Chapter 1).

Subsequently, we delve into the nested quantum search algorithm developed by Cerf et al.

(2000, hereafter referred to as NQC). Serving as the quantum counterpart to classical nested search algorithms, NQC facilitates the identification of a valid solution within a set of 2^n solutions using only $O(\sqrt{2\gamma n})$ Grover iterations, where γ is a parameter ranging from 0 to 1, contingent on the problem's structural characteristics. Applying NQC to a discrete optimization problem, we observe two significant challenges: a low probability of finding the optimal solution and potential constraints imposition issues during a nested search. Regrettably, our analysis indicates that these issues remain unresolved, and a nested quantum search is outperformed by a classical nested search employing, for instance, GUM to identify (partial) solutions to nested problems.

Finally, we turn our attention to amplitude amplification, a broader concept encompassing Grover's algorithm. Amplitude amplification aims to minimize the number of iterations and operations required. While Grover's algorithm initiates with a "uniform superposition" of qubits, amplitude amplification employs a tailored superposition that enables the measurement of a valid solution with considerably fewer iterations. However, determining an optimal superposition is non-trivial. We demonstrate that, adhering to logical rules, it is possible to arrive at a suboptimal superposition that performs worse than a uniform superposition.

Discrete optimization: Limitations of existing quantum algorithms

Stefan Creemers*
IESEG School of Management, Lille, France, s.creemers@ieseg.fr
KU Leuven, Leuven, Belgium, stefan.creemers@kuleuven.be

Pérez Luis Fernando
IESEG School of Management, Lille, France, l.perezarmas@ieseg.fr

We investigate the limitations of existing quantum algorithms to solve discrete optimization problems. First, we discuss the quantum counting algorithm of Brassard et al. (1998), and show that it has performance that is equivalent to that of a brute-force approach when approximating the number of valid solutions. In addition, we show that a straightforward application of Grover's algorithm (referred to as `GUM` by Creemers and Pérez (2023b)) dominates any quantum counting algorithm when verifying whether a valid solution exists. Next, we discuss the nested quantum search algorithm of Cerf et al. (2000), and show that it is dominated by a classical nested search that uses an approach such as `GUM` to find (partial) solutions to (nested) problems. Last but not least, we also discuss amplitude amplification (a procedure that generalizes Grover's algorithm), and show (once more) that it may not be possible to outperform `GUM`.

Keywords: Quantum; computing; algorithm; counting; nested; amplification.

1. Introduction

In this paper, we discuss the limitations of a number of well-known quantum algorithms. First, we discuss the quantum counting algorithm of Brassard et al. (1998; hereafter referred to as `QCB`). The goal of a quantum counting algorithm is to approximate the number of valid solutions (m) in a set of 2^n solutions. An (accurate) approximation of m is particularly useful when using Grover's algorithm: using m, we can determine the number of iterations (required by Grover's algorithm) that maximizes the probability to measure any of the m valid solutions. In addition, quantum counting algorithms can also be used to evaluate whether a solution exists (i.e., to approximate $P(m > 0)$). In what follows, however, we show that `QCB` requires 2^n Grover iterations to accurately approximate m (this is equivalent to the performance of a brute-force classical approach). In addition, we show that any quantum counting algorithm (i.e., not only `QCB`) is dominated by Algorithm 1 (hereafter referred to as `GUM`) when assessing whether $m > 0$.

Procedure `GUM` is a straightforward application of Grover's algorithm that may be used to assess whether a valid solution exists in a set of 2^n solutions if m is unknown (Creemers and Pérez, 2023b).

*Corresponding author

Because m is unknown, GUM evaluates several assumed numbers of valid solutions ($\widehat{m}$). Starting with $\widehat{m} = 2^n$, GUM calls Grover's algorithm, and performs $I(n, \widehat{m}) = \left\lceil \pi 4^{-1} \sqrt{m^{-1} 2^n} \right\rceil$ Grover iterations (see also Grover (1996)). In each Grover iteration, a function $f_{\mathbf{x}}$ (that returns 1 if solution $\mathbf{x}$ is valid and 0 otherwise) is called twice (once to determine whether solution $\mathbf{x}$ is valid and once to uncompute; refer to e.g., Nielsen and Chuang (2010) for more details). If, after $I(n, \widehat{m})$ Grover iterations, a valid solution $\mathbf{x}$ is found, GUM stops and returns $\mathbf{x}$. If no valid solution is found, we let $\widehat{m} = \widehat{m}/2$ and try again. If no solution is found after processing $\widehat{m} = 1$, GUM stops, and we conclude that no valid solution exists (i.e., $m = 0$). The number of times GUM expects to call function $f_{\mathbf{x}}$ is given by:

$$\sum_{i=0}^{n} P(n, m, 2^{(n-i)}) \left((i+1) \sum_{j=0}^{i} 2I(n, 2^{(n-i)}) \right), \tag{1}$$

where $P(n, m, I(n, \widehat{m}))$ is the probability to measure any of the m valid solution if we use $I(n, \widehat{m})$ Grover iterations to search a set of 2^n solutions. One can verify that, in the worst case, GUM requires $O(\sqrt{2^n})$ function calls. If, however, there are m valid solutions, GUM requires $O(\sqrt{m^{-1} 2^n})$ function calls (see Creemers and Pérez (2023b) for a discussion).

Algorithm 1: Procedure that uses Grover's algorithm to find a valid solution when the number of valid solutions is unknown in a system that has n binary decision variables, and where $f_{\mathbf{x}}$ returns 1 if solution $\mathbf{x}$ is valid (and 0 otherwise).

```
procedure GUM(n)
    m̂ = 2^n;
    do
        perform I(n, m̂) iterations of Grover's algorithm equipped with f_x;
        measure basis state |x⟩;
        if f_x = 1 then
            return x;
        else if m̂ > 1 then
            m̂ = m̂/2;
        else
            return ∅;
    while true;
```

GUM can be used as a subroutine in a procedure that solves a discrete optimization problem. For instance, Creemers and Pérez (2023a) use GUM in a binary search procedure (hereafter referred to as BSP) that solves the binary knapsack problem. Algorithm 2 provides an outline of procedure BSP. Procedure BSP uses a binary search procedure that performs L iterations, where L is a logarithmic function of the difference between $V_{\min}$ and $V_{\max}$ (i.e., the upper and lower bounds on optimal solution value V^*, respectively). In each iteration, BSP uses GUM to assess whether a valid solution exists that satisfies the constraints of the optimization problem, and that has a value of at least $V = \lfloor 0.5(V_{\min} + V_{\max}) \rfloor$. If a solution $\mathbf{x}$ is found for which $f_{\mathbf{x}} = 1$, we update the best-found solution, and let $V_{\min} = (V + 1)$. If GUM is unable to find a valid solution, we let $V_{\max} = (V - 1)$.

This process continues until $V_{\min} > V_{\max}$.

Algorithm 2: Binary search procedure (`BSP`) to solve the binary knapsack problem (Creemers and Pérez, 2023a).

```
initialize x* = ∅, V* = 0, V_min, V_max, and start procedure BSP(n, V_min, V_max);
procedure BSP(n, V_min, V_max)
    do
        V = ⌊0.5(V_min + V_max)⌋;
        x = GUM(n);
        if x ≠ ∅ then
            update best-found solution x* = x and its value V* = V;
            V_min = V + 1;
            break;
        else
            V_max = V − 1;
            break;
    while V_min ≤ V_max;
```

Next, we discuss the nested quantum search algorithm of Cerf et al. (2000; hereafter referred to as `NQC`). `NQC` is the quantum equivalent of a classical nested search algorithm, and allows to find a valid solution in a set of 2^n solutions using only $O(\sqrt{2^{\gamma n}})$ Grover iterations, where γ is some number less than or equal to 1 that depends on the structure of the problem. We use `NQC` to solve a discrete optimization problem, and find that: (1) there may only be a low probability to find the optimal solution, and (2) it may not be possible to impose all constraints when performing a nested search. Unfortunately, we show that these problems cannot be resolved, and that a nested quantum search is dominated by a classical nested search that uses e.g., `GUM` to find (partial) solutions to (nested) problems.

Last, but not least, we also discuss amplitude amplification. Amplitude amplification is a generalization of Grover's algorithm that tries to reduce the number of iterations (and hence operations). Whereas Grover's algorithm starts with a "uniform superposition" of qubits (i.e., a superposition where each qubit has an equal probability to collapse into either $|0\rangle$ or $|1\rangle$; a superposition where every solution has equal probability) and requires $\pi 4^{-1}\sqrt{m^{-1}2^n}$ iterations, amplitude amplification uses a superposition that allows to measure a valid solution using (far) less iterations. This is illustrated in Figure 1 that illustrates the uniform superposition (used by Grover's algorithm), a "good" superposition, and a "bad" superposition. A good superposition uses information of the optimization problem to reduce the number of iterations required to measure a valid solution. Finding such a good superposition, however, may not be that easy. In fact, we show that, following logical rules, it is possible to end up with a bad superposition that performs worse than a uniform superposition.

In what follows, Sections 2 and 3 discuss `QCB` and `NQC`. Section 4 discusses the limitations of amplitude amplification. Section 5 concludes. Note that, we also use the aforementioned quantum

Figure 1. Comparison of a uniform superposition (used by Grover's algorithm), a "good" superposition, and a "bad" superposition when measuring a valid solution in a system that has 1 valid solution and n qubits.

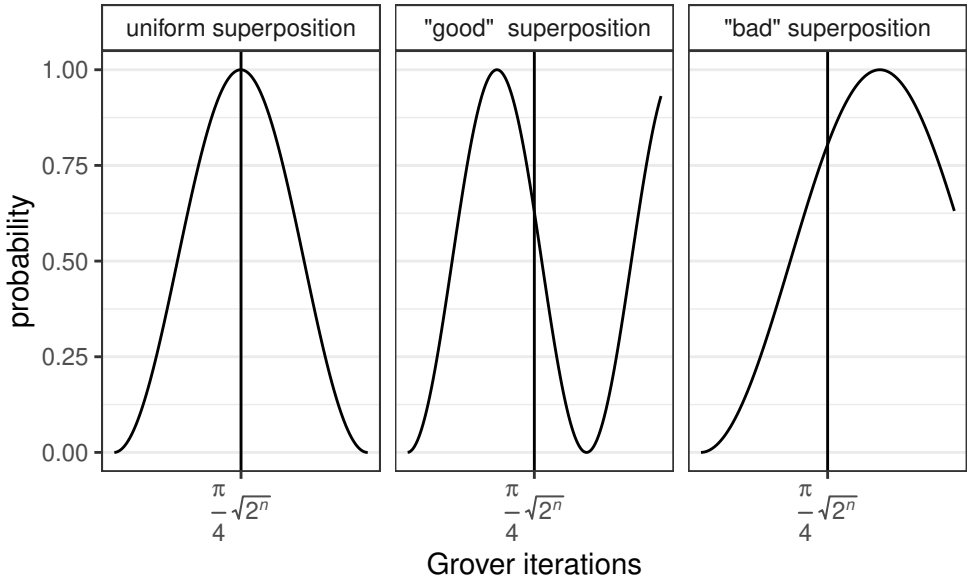


Table 1. Data for example knapsack problem that has $n = 3$ items.

i	w_i	v_i
1	2	3
2	3	1
3	2	2
$W = 4$		

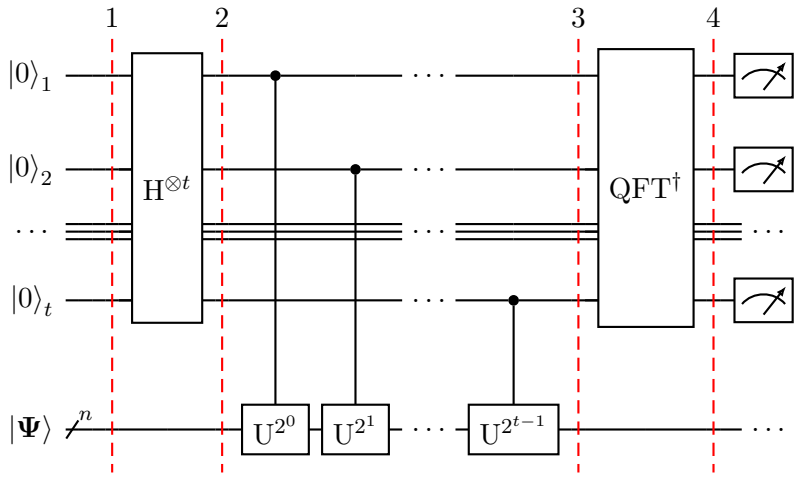
algorithms to solve the binary knapsack problem. We use the example binary knapsack problem that was also used in Creemers and Pérez (2023a and 2023b). The parameters of this example are provided in Table 1, where w_i and v_i denote the weight and value of item i, and W is the maximum weight capacity of the knapsack. The optimal solution of the example knapsack problem is $\mathbf{x}^* = \{1, 0, 1\}$, and has value $V^* = 5$.

2. Quantum counting algorithms

In 1998, Brassard et al. published a quantum counting algorithm (`QCB`) to approximate the number of valid entries in a database that has 2^n entries. In theory, `QCB` can be used to approximate the number of valid solutions (and hence the required number of Grover iterations) before we run Grover's algorithm. Unfortunately, however, `QCB` has a major problem. To illustrate this problem, we first need to introduce the quantum phase estimation algorithm (hereafter referred to as `QPE`).

`QPE` is used as a subroutine in many quantum algorithms, and is considered to be one of the most important quantum algorithms (Kitaev, 1995). The circuit of `QPE` is presented in Figure 2. In this circuit, $|\Psi\rangle$ represents the state of a quantum system that has n qubits, U is a unitary

Figure 2. Circuit of `QPE`



operator that operates on quantum system $|\mathbf{\Psi}\rangle$, and $\text{QFT}^\dagger$ is an operator that performs the inverse quantum Fourrier transform on a set of t counting qubits that are used to store the output of `QPE`. The goal of `QPE` is to estimate the change in the phase of quantum system $|\mathbf{\Psi}\rangle$ if we apply operator U on $|\mathbf{\Psi}\rangle$. The resulting estimate is stored in a set of t counting qubits, and the precision of the estimate is determined by the size of t.

`QPE` can be broken down in four steps (indicated by the red dashed lines in the circuit). In what follows, we focus on the third step of `QPE` (this allows us to expose the problem of `QCB`; refer to Appendix A.1 for a complete description of `QPE` as well as a detailed account of how `QCB` is used to solve the example binary knapsack problem). In the third step, for each counting qubit $1 \leq i \leq t$, we perform a controlled $\text{U}^{2^{i-1}}$ operation on the n qubits of quantum system $|\mathbf{\Psi}\rangle$ (i.e., operation $\text{U}^{2^{i-1}}$ is performed if counting qubit i is in basis state $|i\rangle = |1\rangle$). In order to obtain $\text{U}^{2^{i-1}}$ efficiently, we can use a method that is known as "exponentiation by squaring". This method allows to obtain operator U^{2^i} efficiently using the following recursive relationship:

$$\text{U}^{2^i} = \text{U}^{2^{(i-1)}}\text{U}^{2^{(i-1)}}, \quad \forall\ 1 \leq i < t. \tag{2}$$

In case of Shor's algorithm, for instance, `QPE` is used as a subroutine to determine the period of a function $f(r) = a^r \pmod N$. For given values of a and N, we call `QPE`, and perform operator U^{2^i} for each $0 \leq i < t$, where operator U^{2^i} performs operation $a^{2^i} \pmod N$. Note that, for all $i > 0$, exponentiation by squaring can be used to obtain U^{2^i} efficiently. For instance, given $a = 7$, $N = 15$, and $t = 4$, `QPE` performs the following operations: $7^1 \pmod{15}$ (for the first counting qubit), $7^2 \pmod{15}$ (for the second counting qubit), $7^4 \pmod{15}$ (for the third counting qubit), and $7^8 \pmod{15}$ (for the last counting qubit). Exponentiation by squaring allows to obtain $7^2 \pmod{15}$ using $7^1 \pmod{15}$, $7^4 \pmod{15}$ using $7^2 \pmod{15}$, and $7^8 \pmod{15}$ using $7^4 \pmod{15}$ (refer to e.g., Nielsen and Chuang (2010) for more details).

In `QCB`, on the other hand, operator U corresponds to a single iteration of Grover's algorithm. As

a result, U^{2^i} corresponds to 2^i Grover iterations. Unfortunately, however, U^{2^i} cannot be obtained using exponentiation by squaring as this would require full knowledge of U; this would require us to know all valid solutions up front (see also Appendix A.2 for a discussion and a proof). Because exponentiation by squaring cannot be used, operator U^{2^i} can only be obtained by effectively performing 2^i Grover iterations. As a result, `QCB` requires $\sum_{i=0}^{t-1} 2^i = (2^t - 1)$ Grover iterations if t counting qubits are used to approximate the number of valid solutions.

To illustrate the dynamics of `QCB`, we use it to approximate the number of valid solutions in a system that has $n = 3$ qubits and one valid solution. The results are presented in Table 2. From Table 2, we can conclude three things. First, the precision of the measurement depends on the number of t qubits. For instance, if $t = 1$, there are only two possible measurements: $m = 0$ and $m = 8$. For $t = 3$, we have $m = 0$, $m = 1.17$, $m = 4$, $m = 6.83$, or $m = 8$. Second, the possible measurements follow a sine pattern; it is easier to approximate a low/high number of valid solutions. As such, depending on the true value of m, more counting qubits may be needed. In this example, to be certain whether $m = 3$, we need at least 5 counting qubits (with 4 counting qubits we can only measure $m = 2.47 \approx 2$ or $m = 4.00$; it is not possible to decide that 3 is the correct measurement with only 4 counting qubits). Third, the measurement is probabilistic, we need approximately $t = n$ counting qubits to be relatively certain to measure the correct number of valid solutions. If $t = n$, however, `QCB` needs $(2^n - 1)$ Grover iterations that each require 2 calls to function $f_{\mathbf{x}}$ (note that a brute-force, classical approach requires 2^n calls to determine the number of valid solutions).

As such, `QCB` does not seem fit to accurately approximate the number of valid solutions. Often, however, we don't need a precise estimate of the number of valid solutions; it suffices to verify whether a valid solution exists. This is illustrated in Table 3 that, for various values of t, presents the probability to measure a valid solution in a system that has n qubits and only 1 valid solution (i.e., the worst-case scenario; if there are more valid solutions, less counting qubits are required to verify whether $m > 0$). Table 3 shows that we may need far fewer than $t = n$ counting qubits to get a good estimate of $P(m > 0)$. In fact, it seems that $t = \lceil n/2 \rceil$ already provides a probability of 0.29 or 0.51 to find out whether a valid solution exists (for even and uneven n, respectively). This may be of particular interest in our setting, where, for a given value of V, we don't necessarily need to know the solution itself; it suffices to know whether a solution exists. This idea is used in Algorithm 3 (hereafter referred to as `CNT`) that uses a binary search procedure similar to `BSP` to solve a discrete optimization problem. Whereas `BSP` uses `GUM` to verify whether a valid solution exists, however, `CNT` uses `QCB` (or any other counting algorithm). `CNT` performs up to $L \approx \lceil \log_2(V_{\max} - V_{\min}) \rceil$ iterations, and in each iteration, we update the current value V, and use a counting algorithm to evaluate whether a solution exists that has a value of at least V. If a solution exists, we update $V_{\min}$. If no solution exists, we update $V_{\max}$. This procedure continues until we have determined that the optimal solution value is either $V_{\min}$ or $V_{\max}$. At this stage, we use `BSP` (and hence `GUM`) to obtain the solution that corresponds to $V_{\max}$ (or $V_{\min}$ if no solution can be found that has solution value $V_{\max}$). In what follows, we use `CNT` to solve the example binary knapsack problem.

Table 2. Probability to measure m valid solutions using `QCB` for various values of t, given a system that has $n = 3$ qubits and $m = 1$ valid solution. The bottom row reports the probability to conclude that $m = 1$.

m	$P(m)$					
	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$
0.00	0.8750	0.4922	0.0077	0.0072	0.0056	0.0016
0.02						0.0034
0.08					0.0138	0.0040
0.17						0.0053
0.30				0.0367	0.0282	0.0082
0.47						0.0156
0.67					0.1574	0.0456
0.91						0.6378
1.17			0.9816	0.9212	0.7085	0.2051
1.46						0.0315
1.78					0.0428	0.0124
2.11						0.0067
2.47				0.0191	0.0147	0.0042
2.84						0.0030
3.22					0.0077	0.0022
3.61						0.0018
4.00		0.4375	0.0068	0.0064	0.0049	0.0014
4.39						0.0012
4.78					0.0036	0.0010
5.16						0.0009
5.53				0.0036	0.0028	0.0008
5.89						0.0007
6.22					0.0023	0.0007
6.54						0.0006
6.83			0.0028	0.0026	0.0020	0.0006
7.09						0.0005
7.33					0.0018	0.0005
7.53						0.0005
7.70				0.0022	0.0017	0.0005
7.83						0.0005
7.92					0.0016	0.0005
7.98						0.0005
8.00	0.1250	0.0703	0.0011	0.0010	0.0008	0.0002
P(m=1)	0.0000	0.0000	0.9816	0.9212	0.8658	0.9200

Table 3. Probability to measure whether a solution exists using `QCB` for various values of t and n, given a system that has $m = 1$ valid solution (values for which $t = \lceil n/2 \rceil$ are indicated in blue).

t	$P(m>0)$									
	$n=3$	$n=4$	...	$n=7$	$n=8$	...	$n=15$	$n=16$	...	$n=31$	$n=32$
1	0.1250	0.0625	...	0.0078	0.0039	...	0	0	...	0	0
2	0.5078	0.2822	...	0.0386	0.0194	...	0.0002	0.0001	...	0	0
3	0.9923	0.7974	...	0.1541	0.0795	...	0.0006	0.0003	...	0	0
4	0.9928	0.9616	...	0.5119	0.2913	...	0.0026	0.0013	...	0	0
5	0.9944	0.9852	...	0.9884	0.7935	...	0.0104	0.0052	...	0	0
6	0.9984	0.9992	...	0.9895	0.9640	...	0.0410	0.0207	...	0	0
7	0.9997	0.9994	...	0.9930	0.9847	...	0.1559	0.0806	...	0	0
8	0.9999	0.9998	...	0.9993	0.9997	...	0.5122	0.2919	...	0	0
9	1	1	...	0.9995	0.9997	...	0.9881	0.7933	...	0	0
10	1	1	...	1	0.9998	...	0.9893	0.9642	...	0.0002	0.0001
11	1	1	...	1	1	...	0.9930	0.9847	...	0.0007	0.0003
12	1	1	...	1	1	...	0.9993	0.9997	...	0.0026	0.0013
13	1	1	...	1	1	...	0.9996	0.9997	...	0.0104	0.0052
14	1	1	...	1	1	...	1	0.9998	...	0.0410	0.0207
15	1	1	...	1	1	...	1	1	...	0.1559	0.0806
16	1	1	...	1	1	...	1	1	...	0.5122	0.2919
17	1	1	...	1	1	...	1	1	...	0.9881	0.7933
18	1	1	...	1	1	...	1	1	...	0.9893	0.9642

Algorithm 3: Procedure used to solve a discrete optimization problem using a quantum counting algorithm

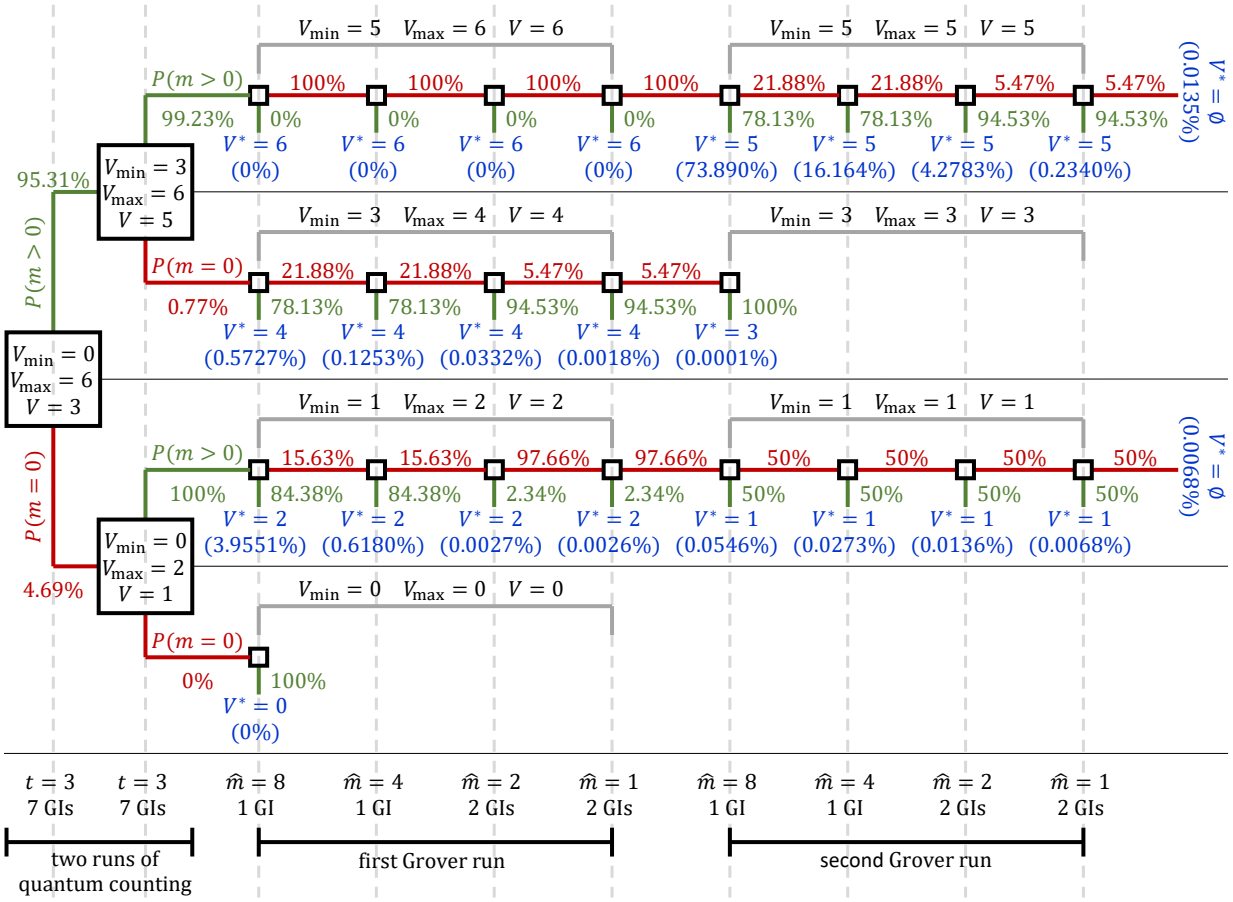
initialize $\mathbf{x}^* = \emptyset$, $V^* = 0$, $V_{\min}$, $V_{\max}$, and start procedure `CNT`$(n, V_{\min}, V_{\max})$;
procedure `CNT`$(n, V_{\min}, V_{\max})$
  do
    $V = \lceil 0.5(V_{\min} + V_{\max}) \rceil$;
    Use quantum counting algorithm to determine $P(m > 0)$ for value V;
    if $P(m > 0) > 0$ **then**
      $V_{\min} = V$;
    else
      $V_{\max} = V - 1$;
  while $(V_{\max} - V_{\min}) > 1$;
  perform `BSP`$(n, V_{\min}, V_{\max})$ to obtain optimal solution $\mathbf{x}^*$ and its value V^*;

Table 4. Probability to measure a solution with value V (or to not measure a valid solution at all; see row "NA"), when using `CNT` to solve the example knapsack problem using various number of counting qubits (t). The optimal solution is printed in blue. The worst-case and expected number of calls to function $f_{\mathbf{x}}$ are also given for each value of t.

V	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$
0	0.3749	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0082	0.0041	0.0010	0.0003	0.0001	0.0000
2	0.3663	0.1831	0.0458	0.0114	0.0029	0.0007
3	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000
4	0.2187	0.3998	0.0073	0.0071	0.0055	0.0016
5	0.0312	0.4125	0.9457	0.9810	0.9914	0.9975
6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
NA	0.0007	0.0003	0.0002	0.0002	0.0001	0.0001
WC[calls]	36	44	60	92	156	284
E[calls]	7.58	20.80	46.76	79.51	143.73	271.86

In the case of our example problem, the dynamics of `CNT` are summarized in Figure 3 for $t = 3$. To solve the example problem using `CNT`, we first initialize the best-found solution, the lower bound $V_{\min} = 0$, and the upper bound $V_{\max} = \sum_{i=1}^{n} v_i = 6$. Next, in a first iteration, we let $V = \lceil 0.5(V_{\min} + V_{\max}) \rceil = 3$, and use `QCB` to verify whether a solution exists that has a value of at least 3. We have a 95.31 percent probability to conclude that a valid solution exists, after which we update the lower bound $V_{\min} = V = 3$, and use the quantum counting algorithm in a second iteration to verify if a solution exists that has value $V = \lceil 0.5(V_{\min} + V_{\max}) \rceil = 5$. If we conclude that a valid solution can be found (with 99.23 percent probability), we use `GUM` to try and find a solution that has value $V = V_{\max} = 6$. No solution can be found that has $V = 6$, and therefore, we have to perform a second run of `GUM` in which we try to find a solution that has value $V = V_{\max} = 5$. In this second run, we are almost certain to measure the optimal solution. If, in the second iteration, we (wrongfully) conclude that no valid solution exists that has $V = 5$ (with 0.77 percent probability), we update the upper bound $V_{\max} = V - 1 = 4$, and perform a first run of `GUM` in which we try to find a solution that has value $V = V_{\max} = 4$. If no solution can be found (with very low probability), we try to find a solution that has value $V = V_{\min} = 3$. In this case, we are 100 percent certain that a valid solution is measured. Returning to the first iteration, if we (wrongfully) conclude that no valid solution exists that has value $V = 3$ (with 4.69 percent probability; see also Figure 3), we update the upper bound $V_{\max} = V - 1 = 2$, and use the quantum counting algorithm in a second iteration to determine whether a solution exists that has a value of at least a value $V = \lceil 0.5(V_{\min} + V_{\max}) \rceil = 1$. In this case, we are 100 percent certain to conclude that a valid solution exists, and we perform a first run of `GUM` in which we try to find a solution that has value $V = V_{\max} = 2$. If no solution can be found (with very low probability), we try to find a solution that has value $V = V_{\min} = 1$. If again no solution exists, we conclude that no solution can be found (with 0.0068 percent probability). In the best case, `CNT` requires 15 Grover iterations (and 31 calls to function $f_{\mathbf{x}}$). In the worst case, 26 Grover iterations are required (and 60 calls). For different values of t, Table 4 summarizes the probability to find the optimal

Figure 3. Illustration of the dynamics of CNT for the example binary knapsack problem. We first perform two runs of QCB with 3 counting qubits (i.e., $t = 3$). Afterwards, depending on the outcome, we perform 1 to 2 runs of GUM and measure the solution. The solution values (and their probabilities) are given in blue. Successful outcomes of the counting algorithm and Grover's algorithm are indicated in green (vice versa, unsuccessful outcomes are indicated in red).



solution, as well as the worst-case and expected number of required function calls (note that BSP has a probability of 99.99 percent to find the optimal solution, and requires at most 35 function calls to solve the example binary knapsack problem; see also Creemers and Pérez (2023a)).

Each run of QCB requires $(2^t - 1) \approx 2^t$ Grover iterations, and each Grover iteration requires 2 calls to function $f_{\mathbf{x}}$. In total, CNT requires $L \approx \lceil \log_2((V_{\max} - V_{\min})) \rceil$ runs of the quantum counting algorithm, after which at most two runs of GUM are required. As a result, CNT requires at most $2L2^t + 2\sum_{i=0}^{n} \left(1 + 2I(n, 2^{n-i})\right)$ calls to function $f_{\mathbf{x}}$ (two calls for each Grover iteration used by the quantum counting algorithm, two calls for each Grover iteration used to measure a valid solution, and one call to validate each of the measured solutions). If we assume $t = n/2$, CNT has the same complexity as BSP, and requires $O(L\sqrt{2^n})$ calls to function $f_{\mathbf{x}}$. If $t > n/2$, however, the performance of CNT degrades, and BSP outperforms CNT. Note that, to ensure a reasonable probability to find the optimal solution, we require t to be larger than $n/2$. In addition, the expected performance of CNT is close to its worst-case performance as CNT always requires at least $2L2^t$ calls (this is also

illustrated in Table 4, where the expected number of calls approaches the worst-case number of calls as t increases; the $2L2^t$ mandatory calls take the upper hand if n and/or t increase).

Note that, recently, several new quantum counting algorithms have been published (see e.g., Suzuki et al. (2020), Aaronson and Rall (2020), and Grinko et al. (2021)). Unfortunately, none of these algorithms allow to accurately approximate $P(m > 0)$ using less than $O(\sqrt{2^n})$ function calls; they are unable to significantly improve upon `QCB` that requires $O(\sqrt{2^n})$ calls to approximate $P(m > 0)$ (assuming $t = n/2$). In fact, in Appendix A.3, we show that it is impossible to accurately approximate $P(m > 0)$ on a quantum computer using less than $O(\sqrt{2^n})$ function calls. We can, however, also use `GUM` to approximate $P(m > 0)$. As shown in Section 1, `GUM` is almost certain to find a solution (if it exists) using at most $O(\sqrt{2^n})$ function calls (i.e., it has optimal performance when approximating $P(m > 0)$). In addition, next to verifying whether $m > 0$, `GUM` also returns a valid solution. Therefore, to verify whether $m > 0$, `GUM` dominates any quantum counting algorithm.

3. Nested quantum search

In this section, we discuss the nested quantum search algorithm (`NQC`) that was introduced by Cerf et al. (2000) in order to improve the performance of Grover's algorithm when solving NP-complete problems. First, however, let's return to the initial purpose of Grover's algorithm: to find a target entry $\mathbf{x}^*$ in an unstructured database that has 2^n entries using a minimum of calls to a function $f_{\mathbf{x}}$ that returns 1 if $\mathbf{x} = \mathbf{x}^*$ (and 0 otherwise). Whereas a classical algorithm requires at most 2^n calls to function $f_{\mathbf{x}}$, Grover's algorithm only needs $\sqrt{2^n}$ calls, resulting in a quadratic speedup. However, if we consider a structured database, Grover's algorithm still needs $\sqrt{2^n}$ calls. A classical algorithm may need far less calls. For instance, if the database is sorted, binary search may be used, and $(n + 1)$ calls suffice to find $\mathbf{x}^*$. In other words, if a classical algorithm can exploit the structure of a problem, it can easily outperform a straightforward implementation of Grover's algorithm.

To address this problem, Cerf et al. (2000) have suggested to use a nested quantum search. Rather than using Grover's algorithm to search the entire solution space to find the optimal solution, a nested quantum search "nests" one quantum search within another. This way, partial solutions (that are obtained efficiently using Grover's algorithm) are used to build other partial solutions that, in turn, can be used to build the optimal solution. The nested quantum search algorithm is the quantum counterpart of a classical nested search algorithm, and often allows to exploit the structure of a problem. A classical nested search algorithm performs $2^{\gamma n}$ operations, where n is the number of binary decision, and γ is some number less-or-equal-than 1 that depends on the problem instance. Cerf et al. (2000) have shown that the nested quantum search only needs $\sqrt{2^{\gamma n}}$ operations, once more resulting in a quadratic speedup.

To illustrate the nested quantum search algorithm, we use it to find a single target entry in a database that has 2^n entries. To keep things simple, we assume a single level of nesting. As a result, the problem of identifying the target entry is divided in two partial problems. In a first partial problem, we try to find the basis state created by the first i qubits that correspond to the

first i bits of the entry. For this purpose, let $\mathbf{x}_{1i}$ represent the first i qubits, and let $f_{\mathbf{x}1i}$ denote the function that returns 1 if $\mathbf{x}_{1i} = \mathbf{x}^*_{1i}$, where $\mathbf{x}^*_{1i}$ are the first i bits of the target entry. Conversely, let $\mathbf{x}_{(i+1)n}$ represent the last $(n-i)$ qubits, and let $f_{\mathbf{x}(i+1)n}$ denote the function that returns 1 if $\mathbf{x}_{(i+1)n} = \mathbf{x}^*_{(i+1)n}$. After solving the first partial problem, we only solve the second partial problem for those basis states where $\mathbf{x}_{1i} = \mathbf{x}^*_{1i}$. The circuit of the nested quantum search algorithm is given in Figure 4, and can be broken down in 5 steps (indicated by the red dashed lines in the circuit). To illustrate the different steps of the algorithm, we use an example that has $n = 3$ qubits (resulting in a database that has 8 entries) and one target basis state (that corresponds to the target entry). The target entry of the database is $\mathbf{x}^* = \{1, 0, 1\}$ (represented by basis state $|101\rangle$). Here, we let $i = 1$ (i.e., in a first subproblem we consider the first qubit, and after having determined the correct basis state for this first qubit, we try to find the correct basis state for the last two qubits). In a first step of the algorithm, we initialize $n + 2$ qubits: a z qubit that ensures that the nested search on the last two qubits is only performed for basis states where $\mathbf{x}_{11} = \mathbf{x}^*_{11} = \{1\}$, the set containing the first x qubit, the set containing the last two x qubits, and a y qubit that is used to induce phase kickback. After step 1, we obtain $|\mathbf{\Psi}_1\rangle = |00001\rangle$. Next, in step 2, all qubits except the z qubit are passed through a Hadamard gate. We obtain $|\mathbf{\Psi}_2\rangle = |0 + + + -\rangle$. In step 3, oracle $O_{f\mathbf{x}11}$ maps $|z\rangle$ to $|z \veebar f_{\mathbf{x}11}\rangle$:

$$|\mathbf{\Psi}_3\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \otimes |{+}{+}\rangle \otimes |-\rangle\,.$$

In step 4, we apply $I(1,1) = 1$ Grover iteration in which: (a) oracle $O_{f\mathbf{x}11}$ flips the phase of basis state $|z1\mathbf{x}_{23}y\rangle$, and (b) the diffusion operator reflects the probability amplitudes about the average probability amplitude (note that gate D represents the diffusion operator; see also Creemers and Pérez (2023a)). We have:

$$|\mathbf{\Psi}_{4.a}\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right) \otimes |{+}{+}\rangle \otimes |-\rangle\,,$$

$$|\mathbf{\Psi}_{4.b}\rangle = \frac{1}{\sqrt{2}}\left(-|00\rangle + |11\rangle\right) \otimes |{+}{+}\rangle \otimes |-\rangle\,.$$

Note that, if we apply Grover's algorithm to a system that only has one x qubit, we always have a 50 percent probability to measure the target basis state, regardless of the number of Grover iterations. In step 5, we perform a controlled run of Grover's algorithm if the z qubit is in state $|1\rangle$. In other words, we only perform Grover's algorithm for basis state $|1\mathbf{x}_{11}\mathbf{x}_{23}y\rangle$ (whereas basis state $|0\mathbf{x}_{11}\mathbf{x}_{23}y\rangle$ remains unchanged). We apply $I(2,1) = 1$ Grover iteration, and obtain (unchanged basis states are indicated in gray):

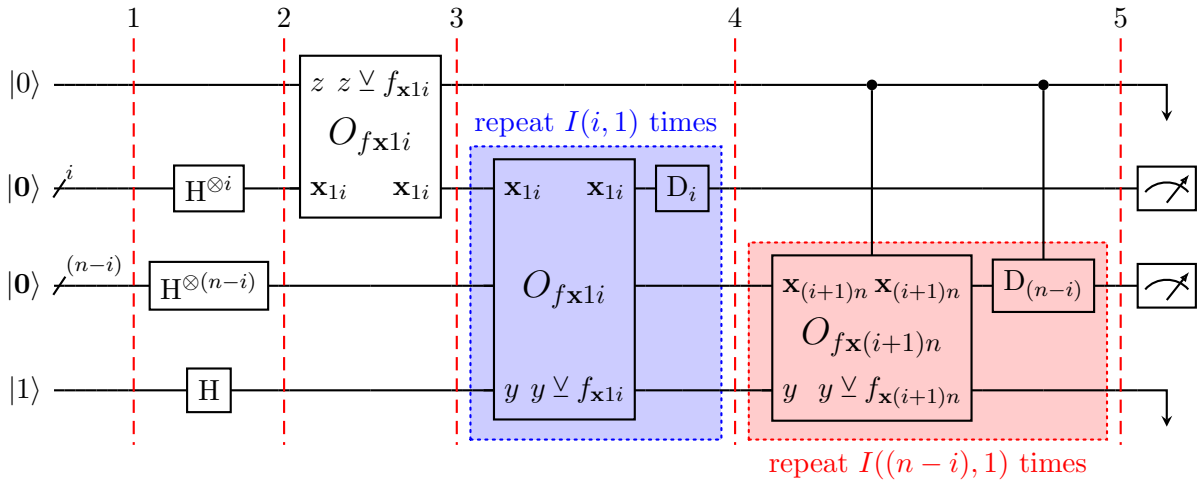
$$|\mathbf{\Psi}_{5.a}\rangle = -\frac{1}{\sqrt{2}}|00{+}{+}{-}\rangle + \frac{1}{\sqrt{2}}|11\rangle \otimes \frac{1}{2}\left(|00\rangle + |10\rangle - |01\rangle + |11\rangle\right) \otimes |-\rangle\,,$$

$$|\mathbf{\Psi}_{5.b}\rangle = -\frac{1}{\sqrt{2}}|00{+}{+}{-}\rangle + \frac{1}{\sqrt{2}}|11\rangle \otimes \left(\frac{2}{2}|01\rangle + \frac{0}{2}(|00\rangle + |10\rangle + |11\rangle)\right) \otimes |-\rangle = -\frac{1}{\sqrt{2}}|00{+}{+}{-}\rangle + \frac{1}{\sqrt{2}}|1101{-}\rangle.$$

Note that, if we apply Grover's algorithm to a system that has two x qubits, we are 100 percent

certain to measure the only target basis state after one Grover iteration. After step 5, we measure the qubits, and are 50 percent certain to measure correct target basis state. If we would have used Grover's algorithm on the entire search space, we need $I(3,1) = 2$ Grover iterations that perform operations on 3 qubits. Using a nested quantum search algorithm, we also require 2 Grover iterations, however, they only perform operations on 1 and 2 qubits, respectively. In addition, note that, for larger n, the nested quantum search algorithm requires far less Grover iterations than a straightforward implementation of Grover's algorithm. That being said, the previous example also allows to identify a number of limitations of the nested quantum search algorithm. In what follows, we discuss these limitations.

Figure 4. Nested quantum search algorithm with one level of nesting.



The computational gain of a nesting algorithm increases with the number of nesting levels. As the number of nesting levels increases, however, the partial problems that need to be solved become smaller and smaller. Unfortunately, for smaller problems, Grover's algorithm is less likely to find the target basis state. In addition, if many partial problems need to be solved, it is likely that, for at least one of them, the target basis state will not be measured. As a result, the target entry (i.e., the "concatenation" of all target basis states) cannot be found. This is also illustrated in our example: we have a 50 percent and 100 percent probability to measure the target basis state in the first and second partial search, respectively. Therefore, the probability to find the target entry in the database only amounts to 50 percent. We conclude that the uncertain outcome of Grover's algorithm (especially when solving small partial problems) limits the computational gains that can be obtained by a quantum nesting algorithm.

Until now, we have assumed that the our database only has a single target entry. If there is only one target entry, each partial search only has one target basis state, and the correct number of Grover iterations can be determined easily. If, on the other hand, the number of target entries is unknown, the number of target basis states in each partial search is also unknown. As a result, to be sufficiently certain that we measure the target basis state in each partial search, we need to evaluate

different numbers of valid solutions for each partial search. For instance, consider a database that has 2^{16} entries. Assume that we have 4 levels of nesting, and that in each of the nested quantum searches we perform Grover's algorithm on 4 qubits. As a result, for each nested search we have 5 possible values for $\widehat{m}$ that need to be verified ($\widehat{m}$ can be 16, 8, 4, 2, and 1), resulting in $4^5 = 1024$ combinations of possible values for $\widehat{m}$ for 4 levels of nesting. As a first combination, we assume $\widehat{m} = 16$ at all four levels; if the target entry was not found, we evaluate a second combination, and assume $\widehat{m} = 16$ at levels 1 to 3, and $\widehat{m} = 8$ at level 4. This process continues until the target entry is found. In the worst case, we need to evaluate 1024 combinations of nested searches to find the target entry. In each of these combinations, we need to perform up to $\sum_{i=0}^{4} I(4, 2^i) = 9$ Grover iterations. In total, a nested quantum search needs at most 9216 Grover iterations. If, on the other hand, we use a straightforward application of Grover's algorithm that searches the entire solution space, we need to perform up to $\sum_{i=0}^{16} I(16, 2^i) = 685$ Grover iterations. As such, if the number of target entries is unknown, a nested quantum search does not necessarily result in a computational gain.

To make matters worse, all of the aforementioned results assume the existence of an efficient function $f_{\mathbf{x}ij}$. Is it possible, however, to implement an efficient function that tells us whether a target basis state of a partial search corresponds to the target entry (i.e., a concatenation of target basis states) without knowledge of the target basis states of other partial searches? Can we tell behind which door we can find the target entry while only observing door handles? In Appendix A.4, we show that it is impossible to implement such a function efficiently, and that `GUM` dominates any nested quantum search algorithm when searching a target entry $\mathbf{x}^*$ in a database that has 2^n entries.

The question remains, however, whether it is possible to use nested quantum search to solve a discrete optimization problem. To answer this question, we first use the nested quantum search algorithm presented in Figure 4 to find a solution to the example binary knapsack problem that has a value of at least $V = 5$ (i.e., $\mathbf{x}^* = \{1, 0, 1\}$ is the only valid solution). In addition, we assume $i = 1$. Therefore, in a first partial problem, we have to decide whether or not to add item 1. We can, however, not use function $f_{\mathbf{x}}$ to evaluate whether or not to add item 1 as $f_{\mathbf{x}}$ will check if a solution where we add item 1: (1) exceeds the weight capacity, and (2) has a value of at least $V = 5$. Because item 1 only has a value of 3, $f_{\mathbf{x}}$ would incorrectly conclude that item 1 is not part of the optimal knapsack solution. As a result, we need to define a "relaxed" version of $f_{\mathbf{x}}$ that does not reject partial solutions that can still be used to build optimal solutions. In this example, other items can still be added in future partial searches such that a value of at least $V = 5$ is obtained; other items, however, cannot reduce the weight of the current knapsack solution (i.e., if the weight of a partial solution exceeds the maximum weight constraint, that partial solution may be rejected). As a result, in a first partial problem we use function $g_{\mathbf{x}}$ that returns 1 if knapsack $\mathbf{x}$ respects the maximum weight constraint. It should be clear that $g_{\mathbf{x}}$ is less effective than $f_{\mathbf{x}}$ in reducing the size of the search space. This is also illustrated in our example, where $g_{\mathbf{x}}$ is unable to decide whether item 1 has to be added (a knapsack with or without item 1 both respect the maximum weight

constraint). We have:

$$|\mathbf{\Psi}_1\rangle = |00001\rangle\,,$$

$$|\mathbf{\Psi}_2\rangle = |0+++-\rangle\,,$$

$$|\mathbf{\Psi}_3\rangle = \frac{1}{\sqrt{2}}\left(|10\rangle + |11\rangle\right) \otimes |{+}{+}\rangle \otimes |-\rangle\,,$$

$$|\mathbf{\Psi}_{4.a}\rangle = \frac{1}{\sqrt{2}}\left(-|10\rangle - |11\rangle\right) \otimes |{+}{+}\rangle \otimes |-\rangle\,,$$

$$|\mathbf{\Psi}_{4.b}\rangle = \frac{1}{\sqrt{2}}\left(-|10\rangle - |11\rangle\right) \otimes |{+}{+}\rangle \otimes |-\rangle\,.$$

In a second (and last) partial search, again, we have to use $g_{\mathbf{x}}$ to decide whether or not to add items 2 and 3. We identify three valid partial solutions: not adding any item, adding only item 2, and adding only item 3. After two Grover iterations, we obtain (note that after $I(2,3) = 1$ Grover iteration we are 0 percent certain to measure a target basis state):

$$|\mathbf{\Psi}_{5.1a}\rangle = -\frac{1}{\sqrt{2}}|10\rangle \otimes \frac{1}{2}(-|00\rangle - |10\rangle - |01\rangle + |11\rangle) \otimes |-\rangle - \frac{1}{\sqrt{2}}|11\rangle \otimes \frac{1}{2}(-|00\rangle - |10\rangle - |01\rangle + |11\rangle) \otimes |-\rangle,$$

$$|\mathbf{\Psi}_{5.1b}\rangle = -\frac{1}{\sqrt{2}}|10\rangle \otimes \left(-\frac{0}{2}(|00\rangle + |10\rangle + |01\rangle) + \frac{2}{2}|11\rangle\right) \otimes |-\rangle - \frac{1}{\sqrt{2}}|11\rangle \otimes \left(-\frac{0}{2}(|00\rangle + |10\rangle + |01\rangle) + \frac{2}{2}|11\rangle\right) \otimes |-\rangle,$$

$$|\mathbf{\Psi}_{5.2a}\rangle = -\frac{1}{\sqrt{2}}|10\rangle \otimes \left(\frac{0}{2}(|00\rangle + |10\rangle + |01\rangle) + \frac{2}{2}|11\rangle\right) \otimes |-\rangle - \frac{1}{\sqrt{2}}|11\rangle \otimes \left(\frac{0}{2}(|00\rangle + |10\rangle + |01\rangle) + \frac{2}{2}|11\rangle\right) \otimes |-\rangle,$$

$$|\mathbf{\Psi}_{5.2b}\rangle = -\frac{1}{\sqrt{2}}|10\rangle \otimes \left(-\frac{1}{2}(|00\rangle + |10\rangle + |01\rangle) + \frac{1}{2}|11\rangle\right) \otimes |-\rangle - \frac{1}{\sqrt{2}}|11\rangle \otimes \left(-\frac{1}{2}(|00\rangle + |10\rangle + |01\rangle) + \frac{1}{2}|11\rangle\right) \otimes |-\rangle.$$

After step 5, we have an equal probability to measure any of the solutions. Note that: (1) several solutions exceed the maximum weight capacity, (2) there is only a 1/8 probability to identify the optimal solution, and (3) we actually never impose the constraint that a solution needs to have a value of at least $V = 5$. Unfortunately, in Appendix A.5 we show that these problems cannot be resolved, and that a nested quantum search is dominated by a classical nested search that uses a procedure that applies Grover's algorithm (e.g., `GUM`) to find a (partial) solution to a (nested) problem.

4. Amplitude amplification

As Grover's algorithm requires $\pi 4^{-1}\sqrt{2^n}$ iterations to measure a (single) valid solution, it may be difficult to solve discrete problems using less than $O(\sqrt{2^n})$ operations when using a Grover-based algorithm. To resolve this problem, amplitude amplification tries to reduce the number of iterations (and hence operations). It does so by generalizing Grover's algorithm (see e.g., Brassard et al., (2002)). Whereas Grover's algorithm starts with a "uniform superposition" of qubits (i.e., a superposition where each qubit has an equal probability to collapse into either $|0\rangle$ or $|1\rangle$; a

superposition that can be achieved by passing n qubits initialized as $|\mathbf{0}\rangle$ through a Hadamard gate), amplitude amplification uses a superposition that allows to measure the target state using (far) less iterations. This is illustrated in Figure 1 that illustrates the uniform superposition (used by Grover's algorithm), a "good" superposition, and a "bad" superposition. A good superposition uses information of the optimization problem to reduce the number of iterations required to measure a valid solution. For instance, consider the example problem that has $n = 3$ qubits and target basis state $|101\rangle$. If we use Grover's algorithm, we have initial state $|000\rangle$, superposition state $\mathrm{H}^{\otimes 3}|000\rangle = |+++\rangle$, and a 78.125 percent probability to measure target basis state $|101\rangle$ after flipping the phase of $|101\rangle$ and applying diffusion operator $\left(2\left|\mathrm{H}^{\otimes 3}000\right\rangle\left\langle\mathrm{H}^{\otimes 3}000\right| - I_8\right)$:

$$\frac{20}{8\sqrt{8}}|101\rangle + \frac{4}{\sqrt{8}}\left(|000\rangle + |100\rangle + |010\rangle + |110\rangle + |001\rangle + |011\rangle + |111\rangle\right).$$

If, on the other hand, we use superposition $\left|\mathrm{H}^{\otimes 3}0 + 0\right\rangle$, we have initial state $|0+0\rangle$, superposition state $\left|\mathrm{H}^{\otimes 3}0 + 0\right\rangle = |+0+\rangle$, and a 100 percent probability to measure target basis state $|101\rangle$ after flipping the phase of $|101\rangle$ and applying diffusion operator $\left(2\left|\mathrm{H}^{\otimes 3}0 + 0\right\rangle\left\langle\mathrm{H}^{\otimes 3}0 + 0\right| - I_8\right)$ (i.e., the second qubit is always in state $|0\rangle$):

$$\frac{2}{2}|101\rangle + \frac{0}{2}\left(|000\rangle + |100\rangle + |001\rangle\right).$$

This example shows that amplitude amplification can indeed reduce the number of iterations (and hence operations) required to measure a target basis state. It is, however, also possible to find an example where amplitude amplification increases the number of iterations. For instance, if we use superposition $\left|\mathrm{H}^{\otimes 3}0 - 0\right\rangle$, we have 0 percent probability to measure target basis state $|101\rangle$ (i.e., this superposition always puts the second qubit in state $|1\rangle$, making it impossible to measure target basis state $|101\rangle$). The difficulty, therefore, is to identify a "good" superposition.

Imagine, for instance, the following superposition where the first and third qubit have a higher probability to collapse into basis state $|1\rangle$, whereas the second qubit has a higher probability to collapse into basis state $|0\rangle$:

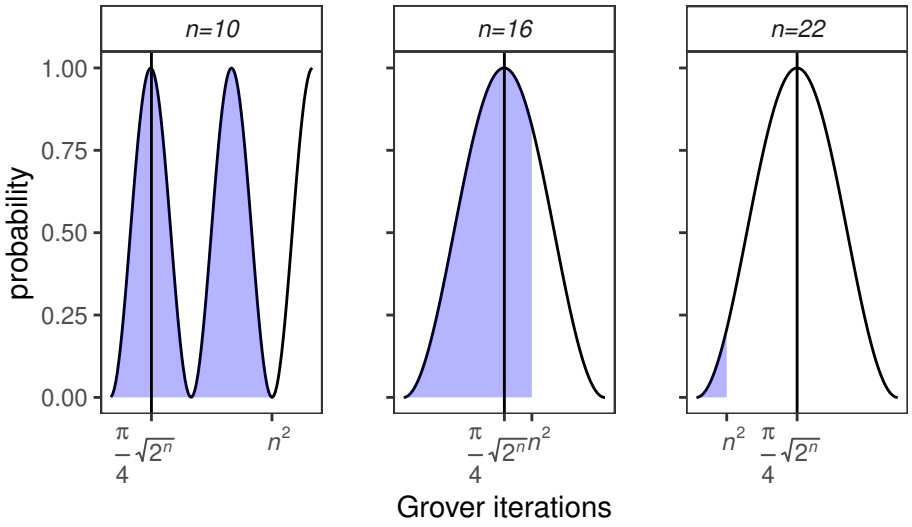
$$\left|\mathbf{\Psi}'\right\rangle = \left(\frac{5\,|0\rangle}{13} + \frac{12\,|1\rangle}{13}\right) \otimes \left(\frac{12\,|0\rangle}{13} + \frac{5\,|1\rangle}{13}\right) \otimes \left(\frac{5\,|0\rangle}{13} + \frac{12\,|1\rangle}{13}\right).$$

Even though this superposition makes practical sense (the qubits have a higher probability to collapse into $|101\rangle$), it does not result in a decreased number of iterations as can be seen in Figure 6 that presents the number of iterations required by a uniform superposition, a "good" superposition ($|\mathbf{\Psi}''\rangle$), and a "bad" superposition ($|\mathbf{\Psi}'\rangle$). From Figure 6, it is clear that superposition $|\mathbf{\Psi}''\rangle$ performs best; it requires the least iterations to measure $|101\rangle$. Superposition $|\mathbf{\Psi}''\rangle$, however, is given by:

$$\left|\mathbf{\Psi}''\right\rangle = \left(\frac{-9\,|0\rangle}{41} + \frac{40\,|1\rangle}{41}\right) \otimes \left(\frac{9\,|0\rangle}{41} + \frac{40\,|1\rangle}{41}\right) \otimes \left(\frac{-9\,|0\rangle}{41} + \frac{40\,|1\rangle}{41}\right).$$

In other words, if we measure superposition $|\mathbf{\Psi}''\rangle$, all qubits have an equal probability (of $40^2/41^2 =$

Figure 5. Number of Grover iterations required to maximize the probability to measure a valid solution in a system that has one valid solution for various values of n. The area in blue is the area that can be sampled if we use up to n^2 Grover iterations.



0.9756) to collapse into basis state $|1\rangle$.

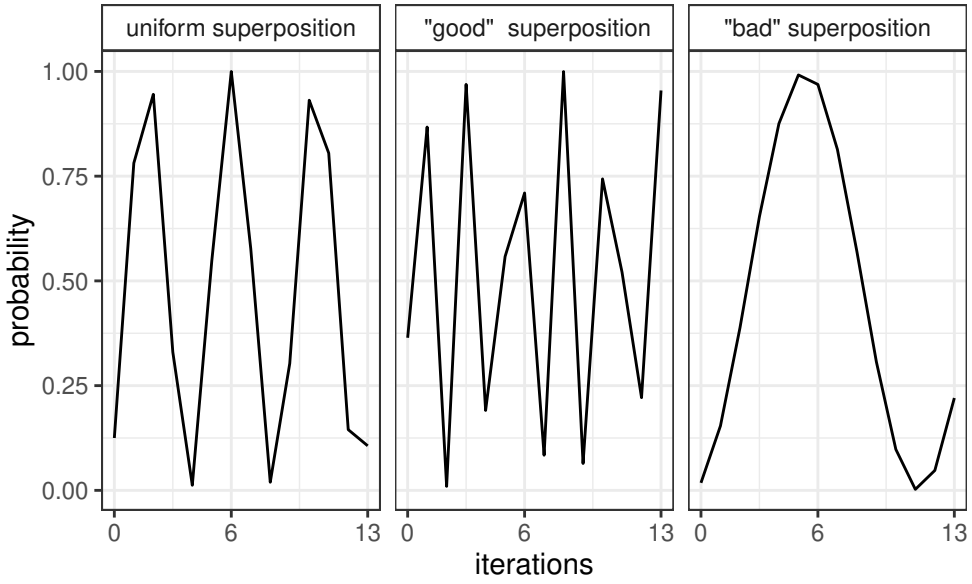
The success of amplitude amplification therefore depends on our ability to find a good superposition based on problem instance characteristics. Note, however, that amplitude amplification cannot be used to perform an unstructured search using less than $O(\sqrt{2^n})$ operations (this would contradict the result of Grover (1996); see also Boyer et al. (1996) and Bennett et al. (1997)). As such, for an unstructured search, amplitude amplification cannot outperform Grover's algorithm. This indicates that it may be difficult for algorithms that use amplitude amplification to outperform Grover-based algorithms (i.e., algorithms that assume a uniform superposition) when solving discrete optimization problems.

5. Conclusion

In this paper, we discuss Grover-based quantum algorithms and investigate their limitations when solving discrete optimization problems.

First, we discuss the quantum counting algorithm of Brassard et al. (1998; referred to as `QCB`). Similar to the prime-factorization algorithm of Shor (1994), `QCB` relies heavily on the quantum phase estimation algorithm (`QPE`). In contrast to the algorithm of Shor, however, `QCB` cannot exploit a technique that is known as "exponentiation by squaring". As a result, `QCB` requires 2^n operations to approximate the number of valid solutions in a set of 2^n solutions (i.e., it has performance equivalent to that of a brute-force approach). In addition, we show that any quantum counting algorithm (i.e., not only `QCB`) is outperformed by a straightforward application of Grover's algorithm (referred to as `GUM`) when assessing whether or not a solution exists. Not only does `GUM` dominate any quantum

Figure 6. Comparison of a uniform superposition (used by Grover's algorithm), a "good" superposition, and a "bad" superposition when measuring a valid solution in a system that has 1 valid solution and 3 qubits.



counting algorithm, it also has optimal performance (i.e., there is no quantum algorithm that has better performance).

Next, we discuss the nested quantum search algorithm of Cerf et al. (2000; referred to as `NQC`). Rather than using Grover's algorithm to search the entire solution space to find the optimal solution, a nested quantum search "nests" one quantum search within another. This way, partial solutions (that are obtained efficiently using Grover's algorithm) are used to build other partial solutions that, in turn, can be used to build the optimal solution. The nested quantum search algorithm is the quantum counterpart of a classical nested search algorithm, and allows to exploit the structure of a problem. Our analysis, however, shows that a nested quantum search algorithm is dominated by a classical (nested) search that uses `GUM` to find solutions to (partial) problems.

Last but not least, we investigate the potential of amplitude amplification. Amplitude amplification is a generalization of Grover's algorithm that tries to reduce the number of iterations (and hence operations) by using a "good" superposition. Finding such a "good" superposition, however, may prove to be difficult. We use amplitude amplification to solve an example problem, and illustrate that the use of logical rules can result in a "bad" superposition (resulting in more iterations than those that are required by Grover's algorithm or in an algorithm that is unable to even find the optimal solution).

A. Appendix

A.1. Quantum phase estimation (QPE) and its use in the quantum counting algorithm of Brassard et al. (1998; QCB)

In this section, we illustrate how QCB can be used to approximate the number of valid binary knapsack solutions. For this purpose, we use the example binary knapsack instance that has 3 items, and assume we are counting the number of solutions that have a value V of at least 5 (i.e., only $\mathbf{x} = \{1, 0, 1\}$ is a valid solution). In addition, we use 3 counting qubits (i.e., $t = 3$).

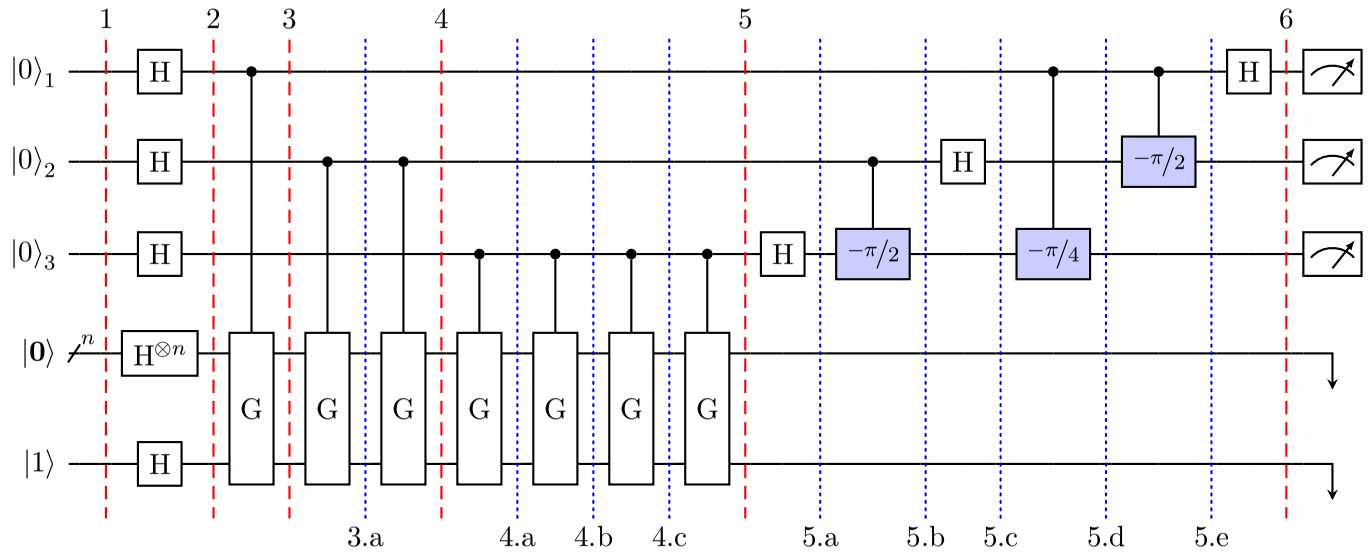
The circuit of the quantum counting algorithm is presented in Figure 7 (whereas red dashed lines indicate the main steps of the algorithm, blue dashed lines indicate substeps). The circuit has three counting qubits, three x qubits representing the binary decision variables, and one y qubit that is used to induce phase kickback. All qubits are initialized as $|0\rangle$ except for the y qubit that is initialized as $|1\rangle$. We have $|\mathbf{\Psi}_1\rangle = |0000001\rangle$. In a first step, all qubits are passed through a Hadamard gate. We obtain $|\mathbf{\Psi}_2\rangle = |+++++-\rangle$. In a second step, we perform a controlled Grover iteration; we perform a Grover iteration if the first counting qubit is in state $|1\rangle$ (i.e., only if the first counting qubit is in state $|1\rangle$, we flip the phase of $|\mathbf{t}101y\rangle$ and apply the diffusion operator). As a result, we have:

$$
\begin{aligned}
|\mathbf{\Psi}_3\rangle = & \frac{|0\rangle}{\sqrt{2}} \otimes |{+}{+}\rangle \otimes \left(\frac{1}{\sqrt{8}} |101\rangle + \frac{1}{\sqrt{8}} (|000\rangle + |100\rangle + |010\rangle + |110\rangle + |001\rangle + |011\rangle + |111\rangle) \right) \otimes |-\rangle \\
& + \frac{|1\rangle}{\sqrt{2}} \otimes |{+}{+}\rangle \otimes \left(\frac{20}{8\sqrt{8}} |101\rangle + \frac{4}{8\sqrt{8}} (|000\rangle + |100\rangle + |010\rangle + |110\rangle + |001\rangle + |011\rangle + |111\rangle) \right) \otimes |-\rangle.
\end{aligned}
$$

In a third step, we perform two Grover iterations controlled by the second counting qubit. After a first Grover iteration, we obtain:

$$
\begin{aligned}
|\mathbf{\Psi}_{3.a}\rangle = & \frac{|00\rangle}{2} \otimes |+\rangle \otimes \left(\frac{1}{\sqrt{8}} |101\rangle + \frac{1}{\sqrt{8}} (|000\rangle + |100\rangle + |010\rangle + |110\rangle + |001\rangle + |011\rangle + |111\rangle) \right) \otimes |-\rangle \\
& + \frac{|10\rangle}{2} \otimes |+\rangle \otimes \left(\frac{20}{8\sqrt{8}} |101\rangle + \frac{4}{8\sqrt{8}} (|000\rangle + |100\rangle + |010\rangle + |110\rangle + |001\rangle + |011\rangle + |111\rangle) \right) \otimes |-\rangle \\
& + \frac{|01\rangle}{2} \otimes |+\rangle \otimes \left(\frac{20}{8\sqrt{8}} |101\rangle + \frac{4}{8\sqrt{8}} (|000\rangle + |100\rangle + |010\rangle + |110\rangle + |001\rangle + |011\rangle + |111\rangle) \right) \otimes |-\rangle \\
& + \frac{|11\rangle}{2} \otimes |+\rangle \otimes \left(\frac{22}{8\sqrt{8}} |101\rangle - \frac{2}{8\sqrt{8}} (|000\rangle + |100\rangle + |010\rangle + |110\rangle + |001\rangle + |011\rangle + |111\rangle) \right) \otimes |-\rangle.
\end{aligned}
$$

Figure 7. `QCB` using $t = 3$ counting qubits to approximate the number of valid solutions in a quantum system that has n qubits. The blue gates represent controlled phase shift gates that shift the phase of the quantum state of a qubit by a period 2π divided by a power of 2.



After a second Grover iteration, we obtain:

$$
\begin{aligned}
|\Psi_4\rangle \;=\; & \frac{|00\rangle}{2}\otimes|+\rangle\otimes\left(\frac{1}{\sqrt{8}}|101\rangle+\frac{1}{\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|10\rangle}{2}\otimes|+\rangle\otimes\left(\frac{20}{8\sqrt{8}}|101\rangle+\frac{4}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|01\rangle}{2}\otimes|+\rangle\otimes\left(\frac{22}{8\sqrt{8}}|101\rangle-\frac{2}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|11\rangle}{2}\otimes|+\rangle\otimes\left(\frac{13}{8\sqrt{8}}|101\rangle-\frac{7}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle.
\end{aligned}
$$

After the third step, all basis states where the first two qubits are in state $|11\rangle$ have undergone three Grover iterations, all basis states where the first two qubits are in state $|01\rangle$ have undergone two Grover iterations, all basis states where the first two qubits are in state $|10\rangle$ have undergone one Grover iteration, and all basis states where the first two qubits are in state $|00\rangle$ have not undergone any Grover iteration. In a fourth step, we perform four Grover iterations controlled by the third counting qubit. After step 4, one can verfiy that the state of the system is given by:

$$
\begin{aligned}
|\Psi_5\rangle \;=\; & \frac{|000\rangle}{\sqrt{8}}\otimes\left(\frac{1}{\sqrt{8}}|101\rangle+\frac{1}{\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|100\rangle}{\sqrt{8}}\otimes\left(\frac{20}{8\sqrt{8}}|101\rangle+\frac{4}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|010\rangle}{\sqrt{8}}\otimes\left(\frac{22}{8\sqrt{8}}|101\rangle-\frac{2}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|110\rangle}{\sqrt{8}}\otimes\left(\frac{13}{8\sqrt{8}}|101\rangle-\frac{7}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|001\rangle}{\sqrt{8}}\otimes\left(\frac{-5}{16\sqrt{8}}|101\rangle-\frac{17}{16\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|101\rangle}{\sqrt{8}}\otimes\left(\frac{-67}{32\sqrt{8}}|101\rangle-\frac{23}{32\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|011\rangle}{\sqrt{8}}\otimes\left(\frac{-181}{64\sqrt{8}}|101\rangle-\frac{1}{64\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|111\rangle}{\sqrt{8}}\otimes\left(\frac{-275}{128\sqrt{8}}|101\rangle+\frac{89}{128\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle.
\end{aligned}
$$

Where basis states with counting qubits in state $|000\rangle$, $|100\rangle$, $|010\rangle$, $|110\rangle$, $|001\rangle$, $|101\rangle$, $|011\rangle$, and $|111\rangle$ have undergone 0, 1, 2, 3, 4, 5, 6, and 7 Grover iterations, respectively. In what follows, we simplify notation, and let $|\mathbf{x}_o\rangle = (|000\rangle + |100\rangle + |010\rangle + |110\rangle + |001\rangle + |011\rangle + |111\rangle)$. As such, $|\Psi_5\rangle$ may be written as:

$$
\begin{aligned}
|\Psi_5\rangle \;=\; & \left(\frac{|000\rangle}{8}+\frac{5|100\rangle}{16}+\frac{11|010\rangle}{32}+\frac{13|110\rangle}{64}-\frac{5|001\rangle}{128}-\frac{67|101\rangle}{256}-\frac{181|011\rangle}{512}-\frac{275|111\rangle}{1024}\right)\otimes|101\rangle\otimes|-\rangle \\
+\; & \left(\frac{|000\rangle}{8}+\frac{|100\rangle}{16}-\frac{|010\rangle}{32}-\frac{7|110\rangle}{64}-\frac{17|001\rangle}{128}-\frac{23|101\rangle}{256}-\frac{|011\rangle}{512}+\frac{89|111\rangle}{1024}\right)\otimes|\mathbf{x}_o\rangle\otimes|-\rangle.
\end{aligned}
$$

In a fifth step, we perform an inverse quantum Fourrier transform on the counting qubits (see also Figure 8 for a general implementation of the circuit of the inverse quantum Fourrier transform; refer to Nielsen and Chuang (2010) for more details). First, the inverse quantum Fourrier transform applies a Hadamard gate to the last counting qubit. As a result, $|t_1t_2\rangle \otimes |0\rangle \otimes |\mathbf{x}\rangle \otimes |-\rangle$ and $|t_1t_2\rangle \otimes |1\rangle \otimes |\mathbf{x}\rangle \otimes |-\rangle$ transform to:

$$\begin{aligned} |t_1t_2\rangle \otimes \mathrm{H}\,|0\rangle \otimes |\mathbf{x}\rangle \otimes |-\rangle &= |t_1t_2\rangle \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes |\mathbf{x}\rangle \otimes |-\rangle\,, \\ |t_1t_2\rangle \otimes \mathrm{H}\,|1\rangle \otimes |\mathbf{x}\rangle \otimes |-\rangle &= |t_1t_2\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \otimes |\mathbf{x}\rangle \otimes |-\rangle\,. \end{aligned}$$

For instance, this implies that $512^{-1}\,|011\rangle \otimes |\mathbf{x}_o\rangle \otimes |-\rangle$ transforms to:

$$\left(\frac{|010\rangle}{512\sqrt{2}} \otimes |\mathbf{x}_o\rangle \otimes |-\rangle\right) - \left(\frac{|011\rangle}{512\sqrt{2}} \otimes |\mathbf{x}_o\rangle \otimes |-\rangle\right).$$

After applying the Hadamard gate we get:

$$\begin{aligned} |\mathbf{\Psi}_{5.a}\rangle = {} & \frac{88|000\rangle{+}52|100\rangle{-}10|010\rangle{-}67|110\rangle{+}168|001\rangle{+}588|101\rangle{+}714|011\rangle{+}483|111\rangle}{1024\sqrt{2}} \otimes |101\rangle \otimes |-\rangle \\ + {} & \frac{-8|000\rangle{-}28|100\rangle{-}34|010\rangle{-}23|110\rangle{+}264|001\rangle{+}156|101\rangle{-}30|011\rangle{-}201|111\rangle}{1024\sqrt{2}} \otimes |\mathbf{x}_o\rangle \otimes |-\rangle. \end{aligned}$$

Next, we apply a controlled phase-shift gate that shifts the phase of the last counting qubit with period $-\pi/2$ if the second counting qubit is in state $|1\rangle$ (in general, the inverse quantum Fourrier transform shifts the phase of a qubit q_t with period $-\pi 2^{(t-t')}$ if control qubit $q_{t'}$ is in state $|1\rangle$; see also Figure 8). A phase-shift gate maps $|0\rangle \rightarrow |0\rangle$ and $|1\rangle \rightarrow e^{i\lambda}\,|1\rangle$, where λ is the period of the shift. As a result, in step 5.a, we map $|0\rangle \rightarrow |0\rangle$ and $|1\rangle \rightarrow -i\,|1\rangle$ (note that $e^{-i\pi/2} = -i$ in step 5.a). We obtain:

$$\begin{aligned} &|\mathbf{\Psi}_{5.b}\rangle = \\ &\left(\frac{88|000\rangle{+}52|100\rangle{-}10|010\rangle{-}67|110\rangle{+}168|001\rangle{+}588|101\rangle}{1024\sqrt{2}} - \frac{714}{1024\sqrt{2}}i|011\rangle - \frac{483}{1024\sqrt{2}}i|111\rangle\right) \otimes |101\rangle \otimes |-\rangle \\ &-\left(\frac{8|000\rangle{+}28|100\rangle{+}34|010\rangle{+}23|110\rangle{-}264|001\rangle{-}156|101\rangle}{1024\sqrt{2}} - \frac{30}{1024\sqrt{2}}i|011\rangle - \frac{201}{1024\sqrt{2}}i|111\rangle\right) \otimes |\mathbf{x}_o\rangle \otimes |-\rangle. \end{aligned}$$

One can verify that, after performing step 5, we obtain $|\mathbf{\Psi}_6\rangle$ as given by the probability amplitudes reported in Table 5. Table 5 also reports the probability to measure any of the basis states as well as the total probability to measure any of the basis states of the three counting qubits ($P_{\mathbf{t}}$). For instance, we have a 0.77 percent probability to measure a system where the three counting qubits are in basis state $|000\rangle$. The question now is: what does this mean? To answer this question, we first determine the rotation angle for each of the basis states of the three counting qubits. The rotation angle is a measure of how much the phase of the quantum state has changed (when compared to

Table 5. Probability amplitude of different basis states and probability to measure any of the basis states after using QCB with $t = 3$ counting qubits to approximate the number of valid solutions in a system that has $n = 3$ qubits and $m = 1$ valid solution. For all basis states of the counting qubits, the table also reports the total probability to measure each of the basis states ($P_{\mathbf{t}}$), the rotation angle of the quantum state (θ), and the corresponding number of valid solutions ($m_{\mathbf{t}}$).

$\|\mathbf{t}\rangle$	$\|\mathbf{t}\rangle \otimes \|101\rangle \otimes \|-\rangle$		$\|\mathbf{t}\rangle \otimes \|\mathbf{x}_o\rangle \otimes \|-\rangle$		$P_{\mathbf{t}}$	$\theta_{\mathbf{t}}$	$m_{\mathbf{t}}$
	amplitude	prob.	amplitude	prob.			
$\|000\rangle$	-0.0321	0.0005	0.0218	0.0072	0.0077	0	0
$\|100\rangle$	0.0031	0.0010	0.0321	0.0001	0.0011	3.1416	8
$\|010\rangle$	$0.0090 + 0.0017i$	0.0028	$0.0338 - 0.0411i$	0.0006	0.0034	1.5708	4
$\|110\rangle$	$0.0090 - 0.0017i$	0.0028	$0.0338 + 0.0411i$	0.0006	0.0034	4.7124	4
$\|001\rangle$	$0.1783 + 0.0213i$	0.2651	$0.0836 - 0.5080i$	0.2257	0.4908	0.7854	1.1716
$\|101\rangle$	$0.0040 - 0.0006i$	0.0013	$0.0324 + 0.0150i$	0.0001	0.0014	3.9270	6.8284
$\|011\rangle$	$0.0040 + 0.0006i$	0.0013	$0.0324 - 0.0150i$	0.0001	0.0014	2.3562	6.8284
$\|111\rangle$	$0.1783 - 0.0213i$	0.2651	$0.0836 + 0.5080i$	0.2257	0.4908	5.4978	1.1716

the phase of its initial state $|\mathbf{0}\rangle$), and can be obtained as follows:

$$\theta_{\mathbf{t}} = \frac{2\pi}{2^t} \sum_{i=1}^{t} t_i 2^{(t-i)}.$$

Using $\theta_{\mathbf{t}}$, we can also determine the number of valid solutions that corresponds to a basis state $|\mathbf{t}\rangle$ using:

$$m_{\mathbf{t}} = 2^n \left(\sin \frac{\theta_{\mathbf{t}}}{2} \right)^2.$$

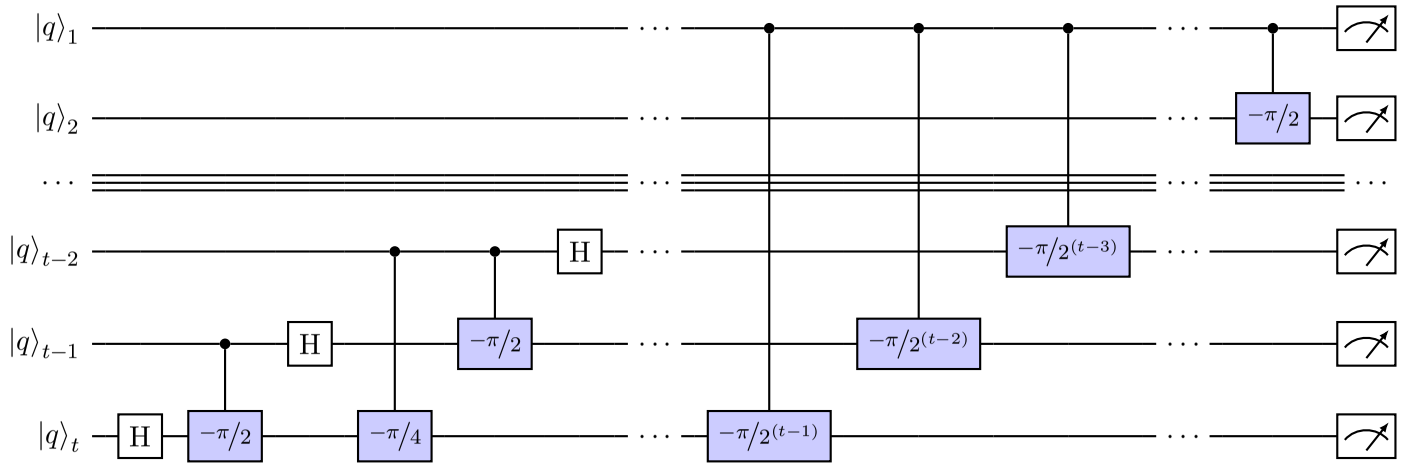
The total probability to measure any of the basis states of the three counting qubits gives us the probability to measure a given number of valid solutions. For instance, we are are 98.16 percent certain to measure there are 1.1716 valid solutions in a system that has 3 qubits (see also Table 2).

A.2. Limitations of exponentiation by squaring

LEMMA 1. Given an operator G that performs a single iteration of Grover's algorithm, exponentiation by squaring cannot be used to obtain G^{2^i} in less than $O(\sqrt{2^n})$ operations, for all $i > 0$.

PROOF. Imagine that we are using Grover's algorithm to find a single target entry $\mathbf{x}^*$ in a database that has 2^n entries. To maximize the probability to find $\mathbf{x}^*$, we need to perform $I(n, 1)$ Grover iterations. Rather than performing $I(n, 1)$ Grover iterations, however, we can use exponentiation by squaring to perform at most $\lceil \log_2(I(n,1)) \rceil \approx \lceil \log_2(\sqrt{2^n}) \rceil = n/2 = t$ operations using t operators G^{2^0}, G^{2^1}, ..., $\mathrm{G}^{2^{(t-1)}}$. For instance, assume $n = 14$. In this case, the corresponding database has 16384 entries, and Grover's algorithm requires $I(14, 1) = 101$ Grover iterations. Rather than performing 101 Grover iterations, we can use exponentiation by squaring, and perform only 4 operations using operators G^{2^0}, G^{2^2}, G^{2^5}, and G^{2^6} (where $2^0 + 2^2 + 2^5 + 2^6 = 101$). In other words, to find $\mathbf{x}^*$, we first need to use exponentiation by squaring to obtain $(n/2 - 1)$ operators G^{2^i} (i.e.,

Figure 8. Circuit of the inverse Quantum Fourrier Transform ($\text{QFT}^\dagger$) using t qubits. The blue gates represent controlled phase shift gates that shift the phase of the quantum state of a qubit by a period 2π divided by a power of 2.



one operator for each $i : 0 < i < n/2$). Next, we perform up to $n/2$ operations using operators G and G^{2^i}. If it takes $\sqrt{2^{(\gamma n)}}$ operations (with $0 < \gamma < 1$) to obtain G^{2^i} (for all $i > 0$), we require $O(n\sqrt{2^{\gamma n}})$ to find $\mathbf{x}^*$. For sufficiently large n, $O(n\sqrt{2^{\gamma n}}) < O(\sqrt{2^n})$. This, however, contradicts the result of Grover (1996), who shows that that at least $O(\sqrt{2^n})$ calls (and hence operations) are required to find a single target entry $\mathbf{x}^*$ in a database that has 2^n entries (see also Boyer et al. (1996) and Bennett et al. (1997)). As a result, even if G is given, we require at least $O(\sqrt{2^n})$ operations to obtain G^{2^i}, for all $i > 0$. □

LEMMA 2. If exponentiation by squaring can be used to obtain G^{2^i} in less than $O(\sqrt{2^n})$ operations (for all $i > 0$), it takes at least $O(\sqrt{2^n})$ operations to obtain operator G (that performs a single iteration of Grover's algorithm).

PROOF. Imagine that we are using Grover's algorithm to find a single target entry $\mathbf{x}^*$ in a database that has 2^n entries. If we use exponentiation by squaring, we need to determine operator G as well as $(n/2 - 1)$ operators G^{2^i} (i.e., one operator for each $i : 0 < i < n/2$). If it takes $\sqrt{2^{(\gamma n)}}$ operations (with $0 < \gamma < 1$) to obtain G^{2^i} (for all $i \geq 0$), we require $O(n\sqrt{2^{\gamma n}})$ operations to find $\mathbf{x}^*$. For sufficiently large n, $O(n\sqrt{2^{\gamma n}}) < O(\sqrt{2^n})$. This, however, contradicts the result of Grover (1996), who shows that that at least $O(\sqrt{2^n})$ calls (and hence operations) are required to find a single target entry $\mathbf{x}^*$ in a database that has 2^n entries (see also Boyer et al. (1996) and Bennett et al. (1997)). As a result, if G^{2^i} can be obtained in less than $O(\sqrt{2^n})$ operations (for all $i > 0$), it takes at least $O(\sqrt{2^n})$ operations to obtain G. □

THEOREM 1. Procedure `GUM` dominates an approach that uses exponentiation by squaring to find a target entry $\mathbf{x}^*$ in a database that has 2^n entries.

PROOF. If we use an approach that uses exponentiation by squaring to find a target entry $\mathbf{x}^*$ in a database that has 2^n entries, Lemma 1 and 2 show that we require a number of operations equivalent to the number of operations that is required to perform $O(\sqrt{2^n})$ calls to a function $f_{\mathbf{x}}$ (that returns 1 if $\mathbf{x} = \mathbf{x}^*$, and 0 otherwise). In addition, Equation 1 shows that `GUM` requires at most $O(\sqrt{2^n})$ calls to measure $\mathbf{x}^*$. As a result, `GUM` dominates any approach that uses exponentiation by squaring to find a target entry $\mathbf{x}^*$ in a database that has 2^n entries. □

A.3. Limitations of quantum counting

THEOREM 2. Given a database that has 2^n entries, there is no quantum algorithm that can accurately approximate $P(m > 0)$ using less than $O(\sqrt{2^n})$ calls to a function $f_{\mathbf{x}}$ that evaluates whether entry $\mathbf{x}$ is the target entry.

PROOF. Imagine that a quantum algorithm exists that can accurately approximate $P(m > 0)$ using $O\sqrt{2^{\gamma n}}$ calls (with $0 < \gamma < 1$). This algorithm can be used in a binary search procedure that identifies a single target entry $\mathbf{x}^*$ in an unstructured database that has 2^n entries using at most $O(log_2(2^n)\sqrt{2^{\gamma n}}) = O(n\sqrt{2^{\gamma n}})$ calls. For sufficiently large n, $O(n\sqrt{2^{\gamma n}}) < O(\sqrt{2^n})$. This, however,

contradicts the result of Grover (1996), who shows that that at least $O(\sqrt{2^n})$ calls are required to find a single target entry $\mathbf{x}^*$ in a database that has 2^n entries (see also Boyer et al. (1996) and Bennett et al. (1997)). As a result, any quantum algorithm that can accurately approximate $P(m > 0)$ requires at least $O(\sqrt{2^n})$ calls. □

To illustrate the binary search procedure used in the proof of Theorem 2, consider a database that has $2^3 = 8$ entries. In a first step of the binary procedure, we evaluate whether target entry $\mathbf{x}^*$ can be found in the first half of the database (i.e., we initialize $|\mathbf{x}\rangle$ as $|0++\rangle$, and use a quantum algorithm to determine whether $m > 0$ for basis states $|000\rangle$, $|001\rangle$, $|010\rangle$, and $|011\rangle$). If $m > 0$, we enter a second step, and verify whether $m > 0$ in the first quarter of the database (i.e., we initialize $|\mathbf{x}\rangle$ as $|00+\rangle$, and evaluate basis states $|000\rangle$ and $|001\rangle$). If, once more, $m > 0$, we enter a third step, and verify whether $|000\rangle$ is the target entry (if it is not, $|001\rangle$ is the target entry). If $m = 0$ in the first quarter, we know $m > 0$ in the second quarter of the database, and verify whether $|010\rangle$ or $|011\rangle$ is the target entry. If $m = 0$ in the first half of the database, we verify whether $m > 0$ in the third quarter of the database (i.e., we initialize $|\mathbf{x}\rangle$ as $|10+\rangle$, and evaluate basis states $|100\rangle$ and $|101\rangle$). If $m > 0$, we verify whether $|100\rangle$ or $|101\rangle$ is the target entry. If $m = 0$, we verify whether $|110\rangle$ or $|111\rangle$ is the target entry. In total, the binary procedure requires $\log_2(2^n) = n$ runs of the quantum algorithm. In each run $i : 1 \leq i \leq n$, we evaluate whether $m > 0$. As such, if a quantum algorithm exists that can approximate $P(m > 0)$ using $O(\sqrt{2^{(\gamma n)}})$ calls, the binary search procedure requires $O(n\sqrt{2^{\gamma n}})$ calls to identify target entry $\mathbf{x}^*$ in an unstructured database that has 2^n entries.

COROLLARY 1. Procedure `GUM` has optimal performance when approximating $P(m > 0)$.

PROOF. From Equation 1 it follows that `GUM` requires at most $O(\sqrt{2^n})$ calls to determine whether $P(m > 0)$; it takes at most $O(\sqrt{2^n})$ calls to measure a target entry (if it exists). In addition, it follows from Theorem 2 that it is not possible to approximate $P(m > 0)$ using less than $O(\sqrt{2^n})$ calls. Therefore, `GUM` has optimal performance when approximating $P(m > 0)$. □

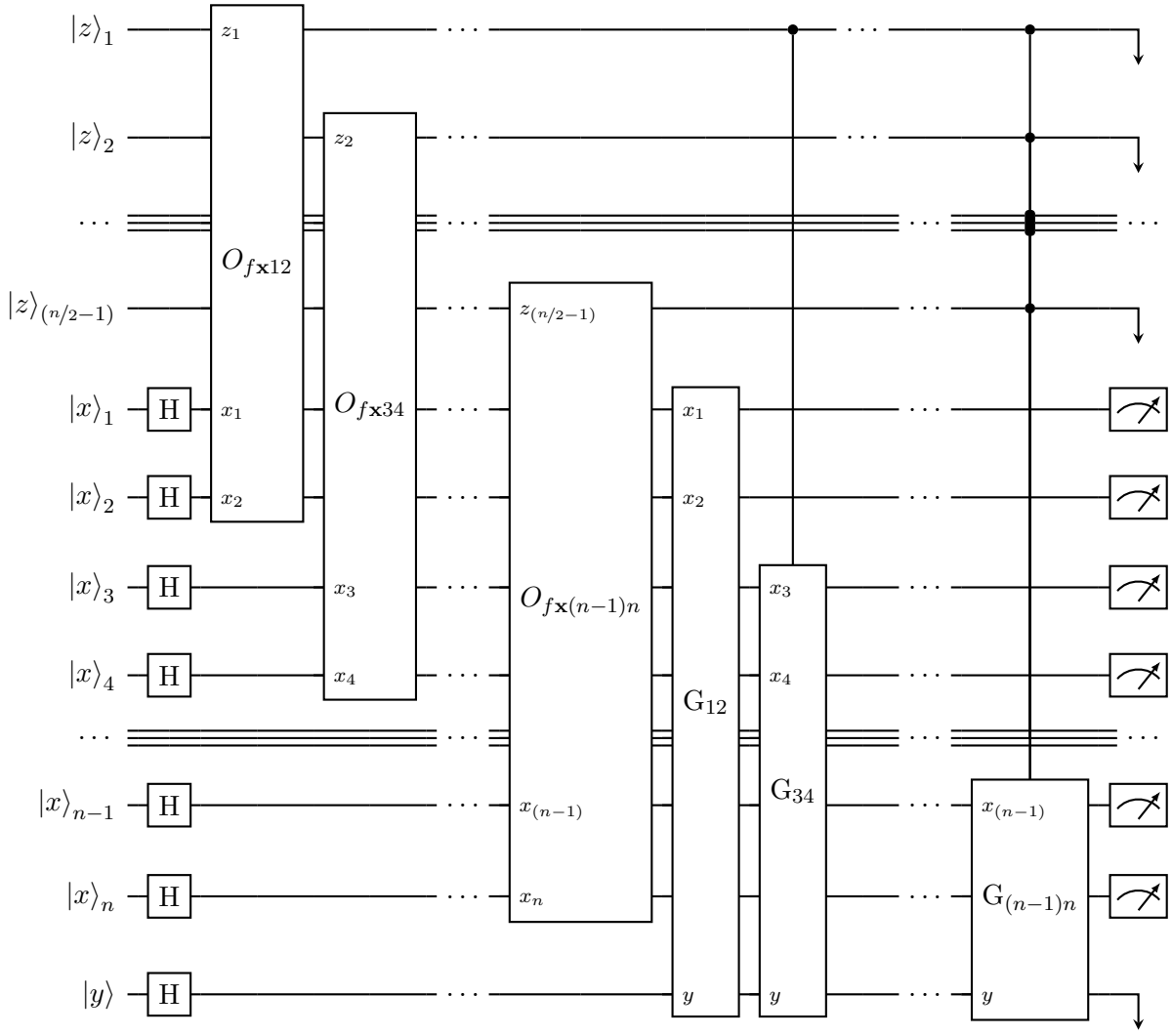
A.4. Limitations of nested quantum search

LEMMA 3. A nested quantum search algorithm can be used to find a single target entry $\mathbf{x}^*$ in a database that has 2^n entries using a linear number of calls to a function $f_{\mathbf{x}i(i+1)}$ that returns 1 if the basis state created by qubits x_i and $x_{(i+1)}$ corresponds to bits i and $(i + 1)$ of target entry $\mathbf{x}^*$, respectively (and 0 otherwise).

PROOF. Without loss of generality, assume that n is an even number. In this case, we can imagine a nested search that has $(n/2 - 1)$ levels of nesting. The circuit of this nested quantum search algorithm is presented in Figure 9. In level i, we solve the partial problem where we search for the basis state created by qubit $(2i - 1)$ and qubit $2i$ that matches bit $(2i - 1)$ and bit $2i$ of the target entry, respectively. In other words, in each level i, we consider two qubits, and are searching for a single target basis state. As a result, in each level i, we require $I(2, 1) = 1$ Grover iteration, and are

100 percent certain to measure the correct basis state (in a system that has two qubits, Grover's algorithm returns the target basis state after 1 Grover iteration with 100 percent probability). In total, we need $n/2$ Grover iterations (that search quantum systems of two qubits) to find the target entry in a database of 2^n entries with certainty, regardless of whether or not this database is structured. Each of these $n/2$ Grover iterations requires 2 calls to function $f_{\mathbf{x}i(i+1)}$. In addition, to prepare the z qubits, we require $(n/2 - 1)$ calls to function $f_{\mathbf{x}i(i+1)}$. As such, a nested quantum search algorithm can find a single target entry $\mathbf{x}^*$ in a database of 2^n entries using a linear number (i.e., $(3n/2 - 1)$) of calls to function $f_{\mathbf{x}i(i+1)}$. □

Figure 9. Nested quantum search algorithm with $(n/2 - 1)$ levels of nesting. Gate $O_{f\mathbf{x}(2i-1)2i}$ represents the oracle that uses function $f_{\mathbf{x}(2i-1)2i}$ to set up the z_i qubit for level i. For each level i, gate $\mathrm{G}_{(2i-i)2i}$ performs a single, controlled Grover iteration on x qubits $x_{(2i-1)}$ and x_{2i}, where qubit y is used to induce phase kickback.



THEOREM 3. A function $f_{\mathbf{x}i(i+1)}$ that returns 1 if the basis state created by qubits x_i and $x_{(i+1)}$

corresponds to bits i and $(i+1)$ of target entry $\mathbf{x}^*$ cannot be implemented on a quantum computer requiring less than $O(\sqrt{2^n})$ operations.

PROOF. It follows from Lemma 3 that we need $O(n)$ calls to function $f_{\mathbf{x}i(i+1)}$ to search a single target entry $\mathbf{x}^*$ in a unstructured database that has 2^n entries. Therefore, if function $f_{\mathbf{x}i(i+1)}$ can be implemented using $O(\sqrt{2^{(\gamma n)}})$ operations (with $0 < \gamma < 1$), it follows that we can find $\mathbf{x}^*$ using less than $O(\sqrt{2^n})$ operations if n is sufficiently large (i.e., $O(n\sqrt{2^{(\gamma n)}}) < O(\sqrt{2^n})$ if n is sufficiently large). This, however, contradicts the result of Grover (1996), who shows that that at least $O(\sqrt{2^n})$ calls (and hence operations) are required to find a single target entry $\mathbf{x}^*$ in a database that has 2^n entries (see also Boyer et al. (1996) and Bennett et al. (1997)). As a result, $f_{\mathbf{x}i(i+1)}$ requires at least $O(\sqrt{2^n})$ operations. □

THEOREM 4. A function $f_{\mathbf{x}i(i+j)}$ that returns 1 if the basis state created by qubits $x_i, x_{i+1}, \ldots, x_{(i+j)}$ corresponds to bits $i, (i+1), \ldots, (i+j)$ of target entry $\mathbf{x}^*$ cannot be implemented on a quantum computer requiring less than $O(\sqrt{2^n})$ operations.

PROOF. The proof of Theorem 4 follows the same logic as the proofs of Lemma 3 and Theorem 3. If $f_{\mathbf{x}i(i+j)}$ can be implemented using $O(\sqrt{2^{(\gamma n)}})$ operations (with $0 < \gamma < 1$), we need $O(\sqrt{2^{(\gamma n)}}) < O(\sqrt{2^n})$ operations to find $\mathbf{x}^*$. This, however, contradicts the result of Grover (1996), who shows that that at least $O(\sqrt{2^n})$ calls (and hence operations) are required to find a single target entry $\mathbf{x}^*$ in a database that has 2^n entries (see also Boyer et al. (1996) and Bennett et al. (1997)). As a result, $f_{\mathbf{x}i(i+j)}$ requires at least $O(\sqrt{2^n})$ operations. □

COROLLARY 2. Procedure `GUM` dominates a nested quantum search that uses function $f_{\mathbf{x}i(i+j)}$ to verify whether the basis state created by qubits $x_i, x_{i+1}, \ldots, x_{(i+j)}$ corresponds to bits $i, (i+1), \ldots, (i+j)$ of target entry $\mathbf{x}^*$.

PROOF. From Theorem 4, for any $j > 0$, it follows that function $f_{\mathbf{x}i(i+j)}$ requires a number of operations that is equivalent to the number of operations required by $O(\sqrt{2^n})$ calls to a function $f_{\mathbf{x}}$ that returns 1 if $\mathbf{x} = \mathbf{x}^*$ (and 0 otherwise). As a result, it follows from Equation 1, that function $f_{\mathbf{x}i(i+j)}$ requires a number of operations that is equivalent to the worst-case number of operations required by `GUM` to find a target entry $\mathbf{x}^*$ in a database that has 2^n entries. Therefore, `GUM` dominates a nested quantum search that uses $f_{\mathbf{x}i(i+j)}$. □

A.5. Limitations of using nested quantum search to solve discrete optimization problems

LEMMA 4. A nested quantum search algorithm can only be used to accurately measure a solution of a discrete optimization problem if all constraints of the optimization problem can be imposed when solving each and every partial problem.

PROOF. If, at any nesting level, a partial problem is solved, solutions that violate constraints that

were not imposed are introduced in the pool of solutions. The probability to measure solutions that violate constraints increases with the number of nesting levels (i.e., there are more opportunities to introduce "bad" solutions) and with the number of constraints that are not imposed (i.e., it becomes more likely that a solution that was introduced is a "bad" solution). □

THEOREM 5. To solve a discrete optimization problem, a nested quantum search algorithm is dominated by a classical nested algorithm that uses GUM to find a (partial) solution for each (nested) problem.

PROOF. We distinguish two cases. In a first case, a nested quantum search algorithm is unable to impose all constraints of the discrete optimization problem at each and every nesting level. In this case, it follows from Lemma 4 that a nested quantum search algorithm may fail to produce the optimal solution. A classical nested search algorithm that uses GUM to search for a (partial) solution, on the other hand, does not suffer from this problem. In a second case, all constraints of the discrete optimization problem can be imposed at each and every nesting level. In this case, however, the discrete optimization problem can be decomposed into independent partial problems for which a solution can be found efficiently using e.g., GUM. We conclude that a nested quantum search algorithm is dominated by a classical nested search procedure that uses GUM to find a (partial) solution for each (nested) problem. □

To further illustrate Lemma 4 and Theorem 5, consider the nested quantum search algorithm presented in Figure 10. In what follows, we use this algorithm to find a binary knapsack solution that has a value of at least V. The new algorithm keeps track of the weight and value of past partial searches, and uses them to decide whether an item can be added when solving the current partial problem. For 2 levels of nesting, the algorithm has 6 steps (indicated by the red dashed lines in the circuit; note that the algorithm can easily be adapted to accommodate more levels of nesting). For a binary knapsack problem that has n items, the algorithm requires $(n-1)$ z qubits, the set of the first i x qubits, the set of $(j-i)$ x qubits, the set of $(n-j)$ x qubits, nb qubits to store the weight of the items, b qubits to store the maximum weight capacity W, b qubits to store the weight of the first partial search, b qubits to store the weight of the second partial search, nb qubits to store the value of the items, b qubits to store the minimum value V, b qubits to store the value of the first partial search, b qubits to store the value of the second partial search, n qubits that store the result of an evaluation function, and 1 y qubit that is used to induce phase kickback (note that we assume b bits are required to represent the integer weights and values used in the binary knapsack problem). The initial state of the system is given by $|\mathbf{\Psi}\rangle = \left|z_1 z_2 \mathbf{x}_{0i}\mathbf{x}_{ij}\mathbf{x}_{jn}\mathbf{c}^w\mathbf{c}^W\mathbf{q}_{0i}^W\mathbf{c}^W\mathbf{q}_{ij}^W\mathbf{c}^v\mathbf{c}^V\mathbf{q}_{0i}^V\mathbf{c}^V\mathbf{q}_{ij}^V y\right\rangle$. To simplify notation, we let $\mathbf{c} = \left\{\mathbf{c}^w, \mathbf{c}^W, \mathbf{c}^v, \mathbf{c}^V\right\}$ represent all constants, and rearrange the qubits as follows: $|\mathbf{\Psi}\rangle = \left|z_1\mathbf{x}_{0i}\mathbf{q}_{0i}^W\mathbf{q}_{0i}^V\right\rangle \otimes \left|z_2\mathbf{x}_{ij}\mathbf{q}_{ij}^W\mathbf{q}_{ij}^V\right\rangle \otimes |\mathbf{x}_{jn}\rangle \otimes |\mathbf{c}\rangle \otimes |y\rangle$.

In what follows, we use the algorithm to find a valid solution to the example binary knapsack problem that has a value of at least $V = 5$ (i.e., only solution $\mathbf{x}^* = \{1, 0, 1\}$ is valid). We have $|\mathbf{\Psi_1}\rangle = |0000\rangle \otimes |0000\rangle \otimes |0\rangle \otimes |23243125\rangle \otimes |1\rangle$ and $|\mathbf{\Psi_2}\rangle = |0++\rangle \otimes |0++\rangle \otimes |+\rangle \otimes |23243125\rangle \otimes |-\rangle$. To simplify notation, let, e.g., $|\mathbf{c}^w\rangle = |010011010\rangle = |232\rangle$, and let $|\mathbf{c}\rangle = |23243125\rangle$ (note that, to-

gether with the y qubit, these qubits remain unchanged for the remainder of the algorithm). Next, we consider the first partial problem, and use function $g_{\mathbf{x}}$ that returns 1 if solution $\mathbf{x}$ does not violate the weight constraint. Note that, function $g_{\mathbf{x}}$ only evaluates the weight constraint; imposing the value constraint could result in the rejection of partial solutions that are used to build an optimal solution (i.e., in the optimal solution, item 1 is added; adding item 1 at this point, however, only leads to a value of 3). In step 2 of the algorithm, we use function $g_{\mathbf{x}}$ to map $|z_1 1\rangle$ to $|11\rangle$ if adding item 1 does not violate the weight constraint (and to $|01\rangle$ otherwise). Conversely, we map $|z_1 0\rangle$ to $|10\rangle$ or $|00\rangle$ if the weight constraint is respected if item 1 is not added (in this case, $|z_1 0\rangle$ always maps to $|10\rangle$). As a result, we obtain:

$$|\mathbf{\Psi}_3\rangle = \left(\frac{|1000\rangle}{\sqrt{2}} + \frac{|1123\rangle}{\sqrt{2}}\right) \otimes \left|z_2 x_2 \mathbf{q}_2^W \mathbf{q}_2^V\right\rangle \otimes |x_3\rangle \otimes |\mathbf{c}\rangle \, |-\rangle \, .$$

Next, we determine the weight and value of the current partial solution. We get:

$$|\mathbf{\Psi}_{3.a}\rangle = \left(\frac{|1000\rangle}{\sqrt{2}} + \frac{|1120\rangle}{\sqrt{2}}\right) \otimes \left|z_2 x_2 \mathbf{q}_2^W \mathbf{q}_2^V\right\rangle \otimes |x_3\rangle \otimes |\mathbf{c}\rangle \, |-\rangle \, ,$$
$$|\mathbf{\Psi}_{3.b}\rangle = \left(\frac{|1000\rangle}{\sqrt{2}} + \frac{|1123\rangle}{\sqrt{2}}\right) \otimes \left|z_2 x_2 \mathbf{q}_2^W \mathbf{q}_2^V\right\rangle \otimes |x_3\rangle \otimes |\mathbf{c}\rangle \, |-\rangle \, .$$

We run Grover's algorithm to increase the probability to measure a valid solution for the first partial problem (note that Grover's algorithm is equipped with function $g_{\mathbf{x}}$ and that $|1000\rangle$ as well as $|1123\rangle$ are valid). After 1 Grover iteration, we get:

$$|\mathbf{\Psi}_4\rangle = \left(-\frac{|1000\rangle}{\sqrt{2}} - \frac{|1123\rangle}{\sqrt{2}}\right) \otimes \left|z_2 x_2 \mathbf{q}_2^W \mathbf{q}_2^V\right\rangle \otimes |x_3\rangle \otimes |\mathbf{c}\rangle \otimes |-\rangle \, .$$

In step 4, we use a function $g_{\mathbf{x},\mathbf{q}}$ that returns 1 if solution $\mathbf{x}$ does not violate the weight constraint given partial solution $\mathbf{q}$. For instance, we use $g_{\mathbf{x},\mathbf{q}}$ to map $|1123\rangle \otimes |z_2 1\rangle$ to $|1123\rangle \otimes |01\rangle$ as adding item 2 to a knapsack that already contains item 1 violates the weight constraint. We get:

$$|\mathbf{\Psi}_{4.a}\rangle = \left(\frac{|1000\rangle}{\sqrt{2}} \otimes \left(\frac{|1000\rangle}{\sqrt{2}} + \frac{|1131\rangle}{\sqrt{2}}\right) + \frac{|1123\rangle}{\sqrt{2}} \otimes \left(\frac{|1023\rangle}{\sqrt{2}} + \frac{|0154\rangle}{\sqrt{2}}\right)\right) \otimes |x_3\rangle \otimes |\mathbf{c}\rangle \otimes |-\rangle \, .$$

At this point, we have a 50 percent probability to either be in a state with $\left|z_1 x_1 \mathbf{q}_1^W \mathbf{q}_1^V\right\rangle = |1000\rangle$ or to be in a state with $\left|z_1 x_1 \mathbf{q}_1^W \mathbf{q}_1^V\right\rangle = |1131\rangle$. In the first case, $\left|z_2 x_2 \mathbf{q}_2^W \mathbf{q}_2^V\right\rangle$ has two valid solutions (i.e., $\left|z_2 x_2 \mathbf{q}_2^W \mathbf{q}_2^V\right\rangle = |1000\rangle$ and $\left|z_2 x_2 \mathbf{q}_2^W \mathbf{q}_2^V\right\rangle = |1131\rangle$). In the second case, $\left|z_2 x_2 \mathbf{q}_2^W \mathbf{q}_2^V\right\rangle$ only has one valid solution (i.e., $\left|z_2 x_2 \mathbf{q}_2^W \mathbf{q}_2^V\right\rangle = |1023\rangle$; $\left|z_2 x_2 \mathbf{q}_2^W \mathbf{q}_2^V\right\rangle = |0154\rangle$ is invalid because it exceeds the maximum weight capacity). As a result, after a single Grover iteration, we obtain:

$$|\mathbf{\Psi}_5\rangle = \left(\frac{|1000\rangle}{\sqrt{2}} \otimes \left(-\frac{|1000\rangle}{\sqrt{2}} - \frac{|1131\rangle}{\sqrt{2}}\right) + \frac{|1123\rangle}{\sqrt{2}} \otimes \left(+\frac{|1023\rangle}{\sqrt{2}} - \frac{|0154\rangle}{\sqrt{2}}\right)\right) \otimes |x_3\rangle \otimes |\mathbf{c}\rangle \otimes |-\rangle .$$

In step 5, we use a function $f_{\mathbf{x},\mathbf{q}}$ that returns 1 if solution $\mathbf{x}$ respects the weight constraint and has

a value of at least $V = 5$ given partial solution $\mathbf{q}$. In this step, Grover's algorithm will only increase the probability of the optimal solution. We get:

$$
\begin{aligned}
|\Psi_6\rangle = &\left(\frac{|1000\rangle}{\sqrt{2}} \otimes \frac{-|1000\rangle}{\sqrt{2}} \otimes \left(\frac{|0\rangle}{\sqrt{2}} + \frac{|1\rangle}{\sqrt{2}}\right)\right) \otimes |\mathbf{c}\rangle \otimes |-\rangle \\
&+ \left(\frac{|1000\rangle}{\sqrt{2}} \otimes \frac{-|1113\rangle}{\sqrt{2}} \otimes \left(\frac{|0\rangle}{\sqrt{2}} + \frac{|1\rangle}{\sqrt{2}}\right)\right) \otimes |\mathbf{c}\rangle \otimes |-\rangle \\
&+ \left(\frac{|1132\rangle}{\sqrt{2}} \otimes \frac{|1032\rangle}{\sqrt{2}} \otimes \left(-\frac{|0\rangle}{\sqrt{2}} + \frac{|1\rangle}{\sqrt{2}}\right)\right) \otimes |\mathbf{c}\rangle \otimes |-\rangle \\
&+ \left(\frac{|1132\rangle}{\sqrt{2}} \otimes \frac{-|0145\rangle}{\sqrt{2}} \otimes \left(\frac{|0\rangle}{\sqrt{2}} + \frac{|1\rangle}{\sqrt{2}}\right)\right) \otimes |\mathbf{c}\rangle \otimes |-\rangle.
\end{aligned}
$$

Because the value constraint is imposed only in the last step, we may be able to increase the probability of measuring basis state $|1132\rangle \otimes |1032\rangle \otimes |1\rangle \otimes |\mathbf{c}\rangle \otimes |-\rangle$ (while reducing the probability of measuring $|1132\rangle \otimes |1032\rangle \otimes |0\rangle \otimes |\mathbf{c}\rangle \otimes |-\rangle$), however, we cannot impact the probability to measure any of the other basis states (that are indicated in gray). As a result, it remains possible to measure a basis state that corresponds to a knapsack solution that has a value $V < 5$ (note that, for large n, there is a low probability to measure a knapsack solution that exceeds the maximum weight capacity because the weight constraint was imposed in all partial problems; in this example, it is still possible to measure a solution that exceeds the maximum weight capacity because each partial problem only involves one item/qubit, resulting in a reduced accuracy of Grover's algorithm). The problems identified in Section 3 can only be resolved if all constraints are imposed when solving each and every partial problem; if the problem can be decomposed into independent partial problems. If, however, we can decompose the main problem into independent partial problems, these partial problems can be solved using a classical nested approach that uses e.g., `GUM` to find solutions to these partial problems.

References

Aaronson, S. and Rall, P. 2020. Quantum approximate counting, simplified. *Symposium on Simplicity in Algorithms*, 24–32.

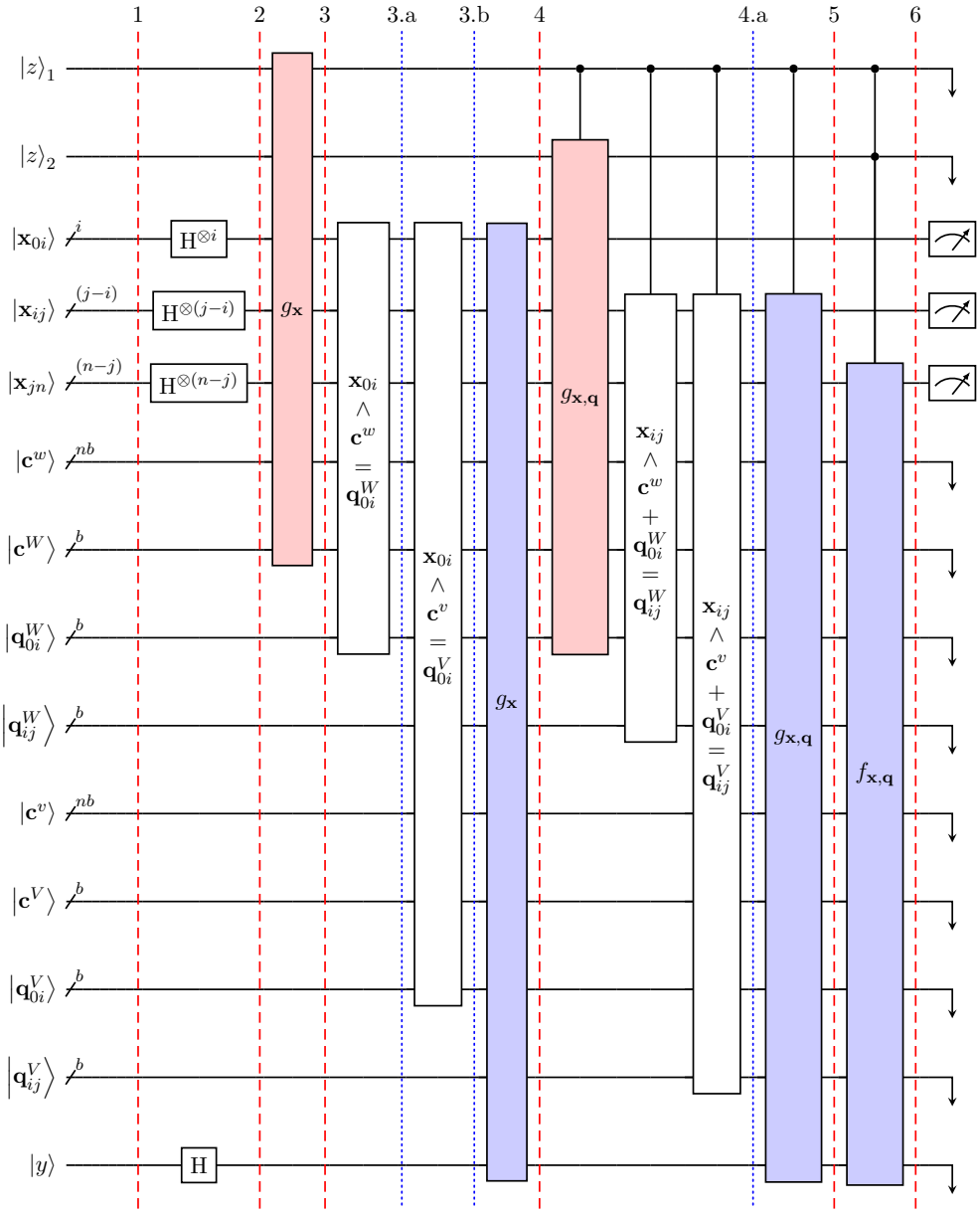
Bennett, C. H., Bernstein, E., Brassard, G., and Vazirani, U. 1997. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, **26**(5), 1510-1523.

Brassard, G., Høyer, P., and Tapp, A. 1998. Quantum counting. In Larsen, K.G., Skyum, S., and Winksel, G. (Eds.), *Automata, Languages and Programming*, 820–831. Springer Berlin Heidelberg, Berlin, Heidelberg.

Brassard, G., Høyer, P., Mosca, M., and Tapp, A. 2002. Quantum amplitude amplification and estimation. In Lomonaco, S.J. and Brandt, H.E. (Eds.), *Quantum computation and information*, 53–74. Amer. Math. Soc., Providence, RI.

Boyer, M., Brassard, G., Høyer, P., and Tapp, A. 1996. *Tight Bounds on Quantum Searching.* Proceedings of the Fourth Workshop on Physics and Computation, New England Complex Systems Institute.

Figure 10. Nested quantum search with two levels of nesting that is used for measuring a binary knapsack solution that has maximum weight W and a value of at least V. Gates that run Grover's algorithm equipped with function $g_{\mathbf{x}}$, $g_{\mathbf{x},\mathbf{q}}$, or $f_{\mathbf{x},\mathbf{q}}$ are indicated in blue. Gates that use functions $g_{\mathbf{x}}$ or $g_{\mathbf{x},\mathbf{q}}$ to initialize z qubits are indicated in red.



Cerf, N. J., Grover, L. K., and Williams, C. P. 2000. Nested quantum search and structured problems. *Phys. Rev. A*, **61**(3), 1–14.

Creemers, S., Pérez, L. 2023a. Discrete optimization: A quantum revolution (Part I), unpublished preprint at http://dx.doi.org/10.2139/ssrn.4198077.

Creemers, S., Pérez, L. 2023b. Discrete optimization: A quantum revolution (Part II), unpublished preprint at http://dx.doi.org/10.2139/ssrn.4494506.

Grinko, D., Gacon, J., Zoufal, C., and Woerner, S. 2021. Iterative quantum amplitude estimation. *npj. Quantum Inf.*, **7**(52).

Grover, L. K. 1996. A fast quantum mechanical algorithm for database search. *Proc. Annu. ACM Symp. Theory Comput.*, 212–219.

Kitaev, A.Y. 1995. Quantum measurements and the Abelian stabilizer problem, unpublished preprint at https://doi.org/10.48550/arXiv.quant-ph/9511026.

Nielsen, M., Chuang, I. 2010. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, Cambridge.

Shor, P.W. 1994. Algorithms for quantum computation: Discrete logarithms and factoring. *Proc. Annu. Symp. FOCS*, 124–134.

Suzuki, Y., Uno, S., Raymond, R., Tanaka, T., Onodera, T., and Yamamoto, N. 2020. Amplitude estimation without phase estimation. *Quantum Inf Process*, **19**(75).

Chapter 4

Quantum discrete optimization: Heuristic approaches for solving the RCPSP

The content of this chapter is presented in the form of a paper titled "**Solving the Resource-Constrained Project Scheduling Problem (RCPSP) with Quantum Annealing**".

4.1 Chapter 4 introduction

This chapter delves into the intricate domain of adiabatic quantum algorithms, specifically focusing on quantum annealing as a tool to address the Resource Constrained Project Scheduling Problem (RCPSP), a formidable NP-Hard scheduling challenge. The choice of RCPSP is grounded in its inherent complexity, wide applicability for minimizing project makespan, the demand for a non-trivial quantum formulation, and the substantial computational cost linked with its quantum annealer-based resolution. Notably, this study pioneers the application of quantum annealing techniques to RCPSP.

The primary objective of this research is to conduct a comprehensive analysis of the capabilities and constraints of existing Quantum Annealing technology. We present a meticulous,

step-by-step methodology for resolving RCPSP using the advanced D-Wave Advantage quantum annealer, boasting an impressive 5640 qubits. Through this exploration, our goal is not only to contribute to the theoretical understanding of quantum annealing but also to showcase its practical utility in addressing real-world optimization challenges.

Following this introduction, the subsequent portions of the paper aim to thoroughly explore both the theoretical underpinnings and practical considerations. The fundamentals of quantum annealing, including the process of mapping problems into Ising and QUBO formulations, are discussed in detail. The architecture of the D-Wave quantum annealer, with a particular focus on the critical aspect of minor embeddings, is elucidated. Benchmarking methodologies, encompassing metrics like Time-to-solution (TTS), Samples-to-target (STT), Time to Target (TTT), and the Atos Q-Score, are introduced to evaluate computational advantages.

Shifting focus to the Resource Constrained Project Scheduling Problem, subsequent sections set the stage for experimental analyses. The QUBO mappings of various Mixed Integer Linear Programming formulations of RCPSP are detailed, presenting the chosen QUBO formulation. An extensive computational experiment in this chapter showcases the performance of quantum annealing for solving the RCPSP, also discussing influential factors such as solution sampling, annealing duration, chain strength, and the implementation of advanced controls like Reverse Annealing and hybrid methods. Finally, the chapter draws conclusions from the comparative performance analysis and highlights open research questions, directing attention to areas necessitating further exploration.

Solving the Resource-Constrained Project Scheduling Problem (RCPSP) with Quantum Annealing

Luis Fernando PÉREZ ARMAS[1*†], Stefan CREEMERS[1,3†] and Samuel DELEPLANQUE[2†]

[1*]Operations Management, IESEG School of Management, Univ. Lille, CNRS, UMR 9221 - LEM - Lille Economie Management, 3 rue de la digue, Lille, F-59800, Nord, France.
[2]Univ. Lille, CNRS, Centrale Lille, Junia, Univ. Polytechnique Hauts-de-France, UMR 8520 - IEMN, 41 Bd Vauban, Lille, 59000, France.
[3]ORSTAT KU Leuven, Naamsestraat 69, Leuven, 3000, Belgium.

*Corresponding author(s). E-mail(s): l.perezarmas@ieseg.fr;
Contributing authors: s.creemers@ieseg.fr; stefan.creemers@kuleuven.be; samuel.deleplanque@junia.com;
†These authors contributed equally to this work.

Abstract

Quantum annealing emerges as a viable solution for solving complex problems such as the resource-constrained project scheduling problem (RCPSP). We analyze 12 Mixed Integer Linear Programming (MILP) formulations for solving the RCPSP, and convert the most qubit-efficient formulation into a Quadratic Unconstrained Binary Optimization (QUBO) model. We solve this QUBO model using the D-Wave Advantage 6.3 Quantum Annealing machine and compare its performance with that of classical computer solvers. This pioneering effort marks the first use of quantum annealing for RCPSP, showing promising results, especially for smaller to medium-sized instances.

Keywords: Resource Constrained Project Scheduling Problem, Quantum Optimization, Quantum Annealing

1 Introduction

In the realm of computational methodologies, quantum computing has emerged as a groundbreaking approach, promising revolutionary solutions to intricate optimization problems. At the forefront of this second quantum revolution is the paradigm of quantum computing, a computational architecture harnessing the principles of quantum mechanics. At the core of quantum computation lies the quantum bit, or qubit, which exhibits unique properties such as superposition and entanglement, paving the way for unprecedented computational capabilities. Currently residing in the NISQ[1] era (Noisy Intermediate-Scale Quantum), quantum computing is marked by limited-size and noise-sensitive quantum machines. The NISQ era acknowledges the ongoing challenges in quantum hardware but emphasizes the potential for valuable research and exploration of quantum advantages, even within the constraints of current technology.

In the landscape of optimization, two prominent approaches dominate the quantum-computing arena. Universal quantum computers, operating using the circuit gate model, with proven advantages in algorithms such as Shor's [2] and Grover's [3], but with limited size (as of 2023 the biggest Universal quantum computer has 433 qubits [4]). On the other hand, adiabatic quantum computers, grounded in the principle of adiabatic quantum computation, exhibit promising experimental results, particularly in optimization problems leveraging the ability to map them as energy minimization problems that allow exploiting the natural tendency of the universe to seek states of minimum energy.

Diverging from gate-based quantum computers, quantum annealers carve a niche with a more focused application scope. These systems boast a higher qubit count (+5000) and enhanced noise resilience, fueling extensive research from both industry and academia. The versatility of quantum annealers is illustrated in their application to complex problems like graph partitioning [5, 6], transportation problems [7, 8], Job-Shop Scheduling Problem (JSSP) [9, 10], and many others applications [11–14].

This paper delves into the realm of adiabatic quantum algorithms, with a specific focus on quantum annealing, to tackle one of the most challenging NP-Hard scheduling problems: the Resource Constrained Project Scheduling Problem (RCPSP). The significance of this choice arises from the inherent complexity of the problem, its broad applicability in projects' makespan minimization, and the non-trivial nature of its formulation as input of the quantum machine. Notably, this work represents the first exploration of applying quantum-annealing techniques to the RCPSP.

The primary objective of this research is to present a comprehensive analysis of the potential and limitations of the current quantum-annealing technology. Our study meticulously outlines a step-by-step approach to solving the RCPSP on the cutting-edge D-Wave Advantage quantum annealer featuring 5640 qubits. Through this exploration, we aim to contribute not only to the theoretical understanding of quantum annealing but also to the practical application of this technology in addressing real-world optimization challenges. Fig. 1 provides a comprehensive visual representation of the methodology and key processes employed in this study.

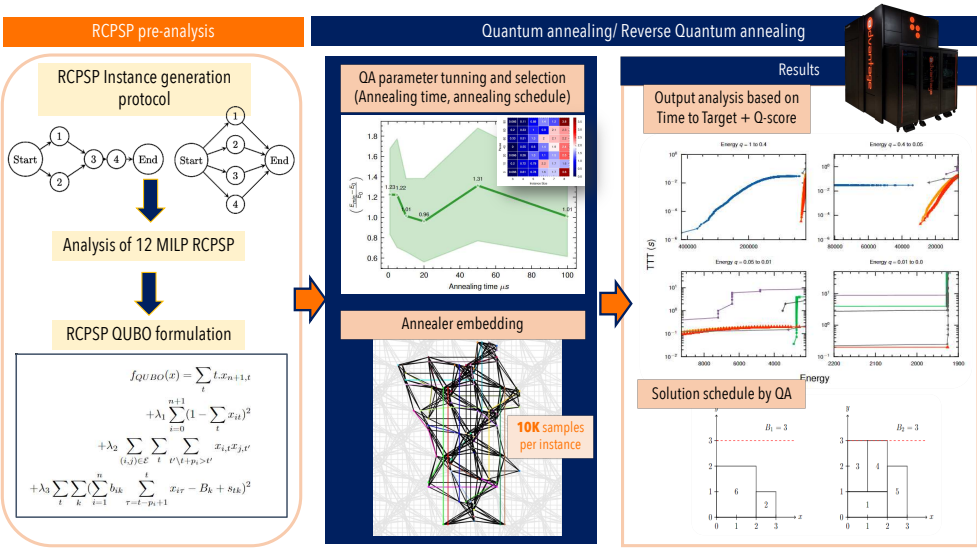


Fig. 1: Visual abstract: a schematic overview of the methodology employed in this study.

This paper provides an in-depth exploration of quantum annealing's theoretical and practical aspects, specifically in relation to the RCPSP. Initially, we examine the quantum-annealing fundamentals, focusing on the problem-embedding process into the annealer's qubit graph. We then introduce the RCPSP and explain the methods used in this study, including the selection of instance generators from existing literature, the adoption of metrics like "Time-to-Target" [15] (TTT) and Atos Q-score, and strategies for utilizing a quantum annealer for RCPSP (annealing time, shots, reverse schedule). The results section presents the first Quadratic Unconstrained Binary Optimization (QUBO) model for RCPSP, derived from an analysis of 12 existing MILP formulations. Subsequently, we conduct an extensive computational experiment to assess the efficacy of quantum annealing when solving the RCPSP and discuss key factors such as solution sampling, annealing duration, chain strength, and the use of advanced controls like reverse annealing. In our conclusions, we also present several promising open research questions.

2 Quantum annealing and D-Wave machines

Quantum Annealing (QA) stands as a quantum metaheuristic designed to tackle combinatorial optimization problems by leveraging the principles of quantum mechanics [16, 17]. This approach draws inspiration from simulated annealing [18], a classical metaheuristic, where a system systematically cools to reach a state of minimum energy. In quantum annealing, quantum phenomena such as superposition and quantum tunneling [19] are harnessed to navigate through local minima efficiently, aiming to pinpoint the global minimum of a cost function. The efficacy of quantum annealing compared to its classical counterpart can vary depending on the specific problem and

hardware employed. The literature offers both theoretical proofs [16], highlighting the advantages of quantum annealing, as well as contrasting perspectives, in particular simulated annealing [20–22].

At the core of quantum annealing lies the profound concept of the natural tendency of the universe to seek systems of minimum energy. This method is grounded in the "adiabatic theorem" of quantum mechanics [23, 24] and the time-dependent Schrödinger equation [25]. It describes the evolution of any quantum system, encapsulating the dynamics of its quantum states. The Hamiltonian, typically represented as a matrix, encapsulates all the information about the quantized energy states available to a quantum system. The adiabatic theorem of quantum mechanics asserts that if a quantum system undergoes a slow and continuous change in its Hamiltonian, it will persist in its instantaneous eigenstate throughout the transformation. In simpler terms, if the system initiates in its ground state and the Hamiltonian changes sufficiently slowly, the system will maintain its ground state.

Applying the adiabatic evolution theorem to optimization problems involves initiating the process with a "simple" initial system represented by the Hamiltonian $\mathcal{H}_0$, for which the ground state can be easily determined. The notations and symbols used throughout this work are reported in Table B1 of appendix B. The system is then gradually transformed into the problem Hamiltonian $\mathcal{H}_1$, an energetic mapping of an optimization problem $\mathcal{P}$. At the conclusion of the adiabatic evolution, the system originally described by $\mathcal{H}_0$ should have transitioned into the ground state of the problem Hamiltonian, representing the minimum value solution to the optimization problem $\mathcal{P}$. The QA algorithm is physically implemented using analog control devices to manipulate a collection of qubit states, following a time-dependent Hamiltonian represented as:

$$\mathcal{H}(t) = A(t)\mathcal{H}_0 + B(t)\mathcal{H}_1$$

This algorithm orchestrates a gradual transition from an initial ground state in $\mathcal{H}_0$ to a state described by the problem Hamiltonian $\mathcal{H}_1$. The $\mathcal{H}_1$ Hamiltonian mirrors the energy function of the optimization problem, ensuring that the ground state for $\mathcal{H}_1$ corresponds to a minimum-cost solution to the optimization problem $\mathcal{P}$. Introduced by Farhi et al. [26], QA and in more general the adiabatic quantum model of computation demonstrates that if the transition is executed slowly enough, the algorithm will, with high probability, converge to a ground state, i.e., an optimal solution.

2.1 Quantum annealing D-Wave implementation

The D-Wave quantum-annealing processors are purposefully engineered for the specific task of identifying minimum-cost solutions to the Ising Minimization problem or, indirectly, to an isomorphic problem: the QUBO. The Ising problem, defined on a graph $G = (V, E)$, entails the assignment of values from $\{-1, +1\}$ to spin variables s_i with the objective of minimizing the following energy function $\mathcal{H}_1$:

$$\mathcal{H}_1 = \sum_i h_i s_i + \sum_{i,j} J_{ij} s_i s_j$$

Where $h = h_i : i \in V$ represent weights, and $J = J_{ij} : (i, j) \in E$ is a set of coupling constants. In the physical context, spin variables s_i can be seen as magnetic poles, with negative J_{ij} indicating ferromagnetic interactions and positive values suggesting antiferromagnetic interactions. The optimal configuration of spin variables s_i that minimizes the energy function is denoted as a ground state, while alternative configurations are classified as excited states that do not have minimum energy. Broadening the computational scope, Ising problems effortlessly transformed into QUBO problems via $s_i = 2x_i - 1$. This modification involves associating binary decision variables $x_i \in \{0, 1\}$ with spin variables $s_i \in \{-1, +1\}$.

A D-Wave Quantum Processor Unit (QPU) maintained at temperatures around a few millikelvin exhibits quantum properties such as superposition and quantum tunneling. Despite the presence of a Faraday shield, the QPU remains susceptible to interference which generally reduces the likelihood of attaining a ground state. Consequently, we categorize any D-Wave processor as a heuristic solver, requiring empirical methods for performance analysis. The current most advanced D-Wave processor (Advantage 6.3) has more than 5000 active qubits, a qubit connectivity of 15 qubits (qubits are not fully connected), and 35000 active couplers, which are made of microscopic loops of niobium. These couplers are connected to a sophisticated analog control system through a network of Josephson junctions [27–29]. Table 1 lists the properties of the different topologies made available by D-Wave: Chimera, Pegasus, and Zephyr.

Topology	Chimera	Pegasus	Zephyr
Device Designation	2000Q	Advantage 6.3	Advantage 2 (Prototype)
Active Qubit Count	2041	5616	563
Connectivity Measure	6	15	20

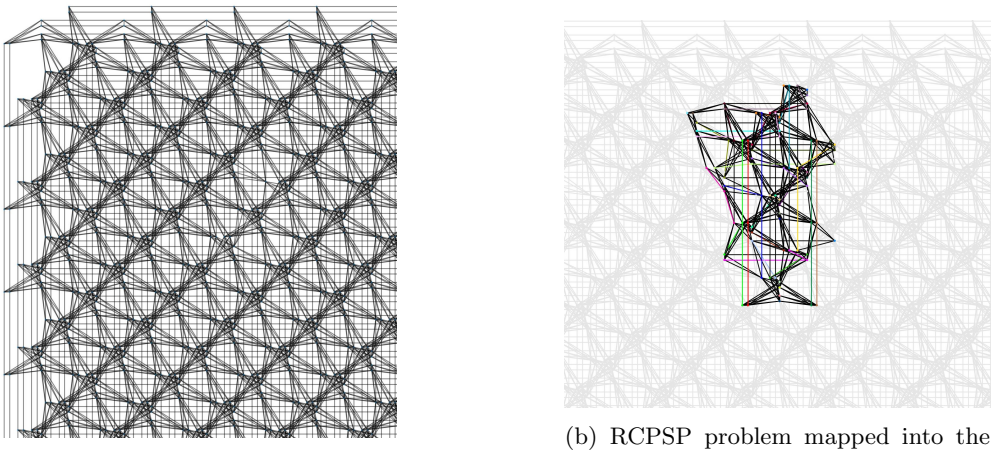
Table 1: Characteristics of available D-Wave quantum devices.

2.1.1 Connectivity and embedding

Besides noise and interference, another major challenge is qubit connectivity. In order to solve a problem on a quantum annealer, the problem graph (QUBO/Ising) must be mapped into the physical hardware, which has a limited number of qubits and interconnections. Consequently, it may be necessary to alter the structure of the QUBO/Ising problem to fit the constrained topology of the quantum annealer. This involves mapping the model onto a larger qubit graph, a critical step that stresses the importance of the machine's inherent topology. This non-trivial process is known as "minor embedding". Heuristics are available in the literature and the search for better embedding algorithms remains an active area of research [30–33]. For this work, minor embeddings are found by using D-Wave's "minor-miner" heuristic [34].

Fig. 2 shows an example of the embedding process with the "Pegasus" Topology. To connect qubits, the minor embedding process creates so-called "chains" that allow to entangle multiple physical qubits. To maintain solution consistency, it is imperative that all physical qubits within a chain have the same value once measured; if not, the chain is considered broken. The chain strength is a parameter introducing a trade-off:

if set too low, it may lead to many broken chains, but if too high, it can impede the qubits' ability to change states.



(a) Pegasus topology of the QPU.

(b) RCPSP problem mapped into the QPU Pegasus topology.

Fig. 2: Minor embedding process with the Pegasus topology.

2.1.2 Anneals and reverse annealing

Once the minor embedding process is successfully completed, the annealing process is initiated. While the adiabatic theorem suggests a potentially lengthy annealing duration to ensure the system stays in the ground state, practical constraints arise due to noise, which can elevate the system to (unwanted) higher energy states. Consequently, a brief annealing is conducted and repeated multiple times in a stochastic process that samples from the energy distribution of the problem. Determining the optimal annealing time and the number of samples depends on the specific problem and its energy distribution. Typical values fall within the range of 10 to 100 microseconds for annealing times and 1 to 10000 for the number of samples.

While the traditional QA starts from the ground state of the "Ising Transverse field" Hamiltonian $\mathcal{H}_0$ and thus from an initial uniform superposition and evolves towards the target problem Hamiltonian $\mathcal{H}_1$, a more sophisticated evolution approach can be applied. In Reverse Annealing (RA), the process departs from an already available solution. Operating in reverse, RA reintroduces partial segments of $\mathcal{H}_0$, thereby reinstating a partial quantum superposition. This unique reverse progression is employed to iteratively refine the initial solution [35–38].

3 Resource-constrained project scheduling Problem

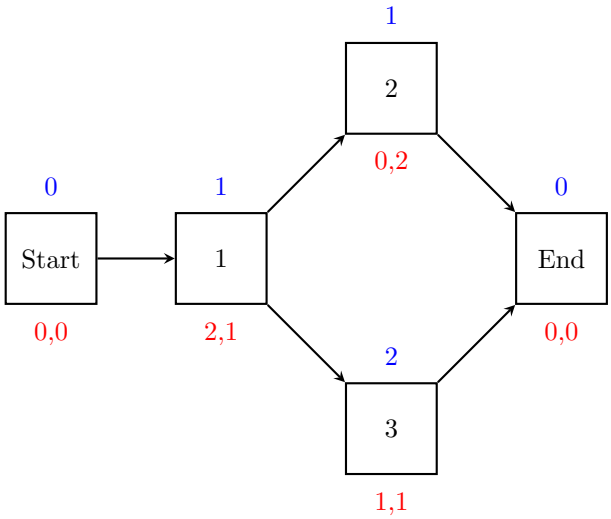
The RCPSP is one of the most extensively studied scheduling problems and perhaps one of the easiest to describe; but despite this apparent simplicity, the RCPSP conceals its true complexity, as demonstrated by Blazewicz et al. [39], who shows that the

RCPSP is NP-hard. This inherent complexity renders the RCPSP one of the most intractable combinatorial optimization problems. Similar to the JSSP [40, 41] the RCPSP falls into the category of problems classified as NP-hard "in the strong sense" [42].

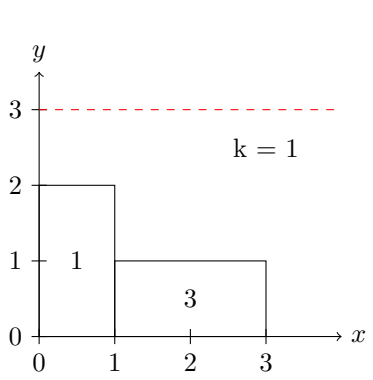
In essence, the RCPSP considers the scheduling of a single project comprising n non-dummy activities, subject to precedence and resource constraints, with the overarching objective of minimizing the project makespan – the total time required for the completion of all activities. The RCPSP is typically represented by a graph $G(\mathcal{A}, \mathcal{E})$ with each node in $\mathcal{A} = \{0, 1, ..., n+1\}$ that corresponds to the different project activities and each edge in $(i, j) \in \mathcal{E}$ equivalent to a straightforward finish-to-start precedence relationship, which means that the start of a successor activity j must await the completion of it's predecessor activity i. Nodes 0 and $n+1$ serve as symbolic milestones, representing the "project start" and "project finish", respectively, these milestone activities are often referred as "dummy activities". Each activity $j \in \mathcal{A}$ has a duration p_j and resource consumption b_{jk}, where k belongs to a set of renewable resource $\mathcal{R}$. Each resource k has a maximum capacity B_k. A feasible solution to the RCPSP corresponds to a project schedule $\mathcal{S} = \{S_0, S_1, \ldots, S_{n+1}\}$ comprised of start times S_j (for each activity $j \in \mathcal{A}$) that respect both precedence and resources constraints. An inherent characteristic of the RCPSP is the non-preemptive nature of activities, indicating that once an activity commences, it cannot be interrupted.

Fig. 3a, depicts the graphical representation of an small RCPSP instance (known as a network diagram) composed of three activities (plus two "dummy" activities) and two renewable resources. The activity duration, denoted as p_j, and resource consumption b_{jk} are presented above and below each node j, respectively. Both renewable resources have a maximum capacity of 3 units. Fig. 3b. shows the optimal schedule $\mathcal{S} = \{S_0 = 0, S_1 = 0, S_2 = 1, S_3 = 1, S_4 = 3\}$ for this instance.

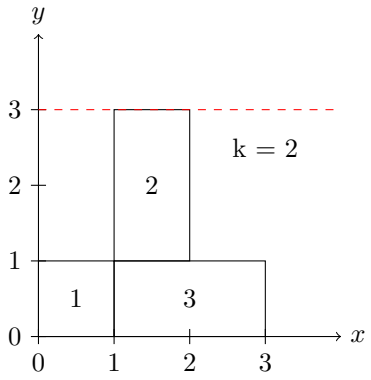
(a) Network diagram.



(b) RCPSP solution schedule $\mathcal{S}$.



(c) Resource profile for resource 1.



(d) Resource profile for resource 2.

Fig. 3: RCPSP instance example: a) RCPSP instance graph $G(\mathcal{A}, \mathcal{E})$ with 3 non-dummy activities and 2 resources, b) solution schedule for the instance, c) resource profile for resource 1, and d) resource profile for resource 2.

3.1 Mixed integer linear programming formulations

Given the intrinsic significance of the RCPSP, it is unsurprising that the literature dedicated to Mixed Integer Linear Programing (MILP) formulations for the RCPSP is

both prolific and dynamic, offering a myriad of approaches. For a given optimization problem, multiple formulations can be devised, they can be distinguished primarily by the formulation size, notably concerning the number of variables, secondly by the number of constraints they involve, and thirdly by the strength of their Linear Programming (LP) relaxation. Typically, for a given problem, a discernible correlation exists between the problem size and the quality of the LP relaxation. Traditionally it has been the case that to enhance the quality of the LP relaxation, a new extended formulation introduces additional variables and constraints, increasing thus the problem size. With the remarkable advancements of LP algorithms, which can now efficiently solve instances involving millions of variables [43–45], it comes as no surprise that, in the context of MILP formulations for the RCPSP, a significant emphasis has been placed on advancing and refining the quality of the LP relaxations [46, 47].

Broadly classified, formulations for the RCPSP fall into three distinct families, as illustrated in Fig. 4. The first category encompasses time-index formulations [48–54], including works by Pritsker et al. 1969 (`PRI69`), Christofides et al. 1987 (`CHR87`), de Sousa and Wolsey 1997 (`SOU97`), Mingozzi et al. 1998 (`MIN98`), Klein and Kaplan 1998 (`KLE98`), Klein 2000 (`KLE00_1` and `KLE00_2`), Demeulemeester and Herroelen 2002 (`DEM02`), and Bianco and Caramia 2013 (`BIA13`). This is followed by sequence-based formulations [55, 56], exemplified by Tamarit and Valdés 1993 (`TAM93`) and Artigues et al. 2003 (`ART03`) . Lastly, the third category comprises event-based formulations [57, 58], including works by Koné et al. 2011 (`KON11`) and Artigues et al. 2013 (`ART13`). An analysis by Koné and Artigues [58, 59] reveals that sequence and event-based formulations tend to be more compact, requiring fewer variables, while time-index formulations, although larger in scale, offer superior Linear Programming (LP) relaxations. It is noteworthy that the majority of time-index formulations tend to be binary formulations, utilizing exclusively binary variables, while event and sequence-based formulations employ a mix between integer and binary variables. For more general information regarding the different MILP formulations of the RCPSP, please refer to [60].

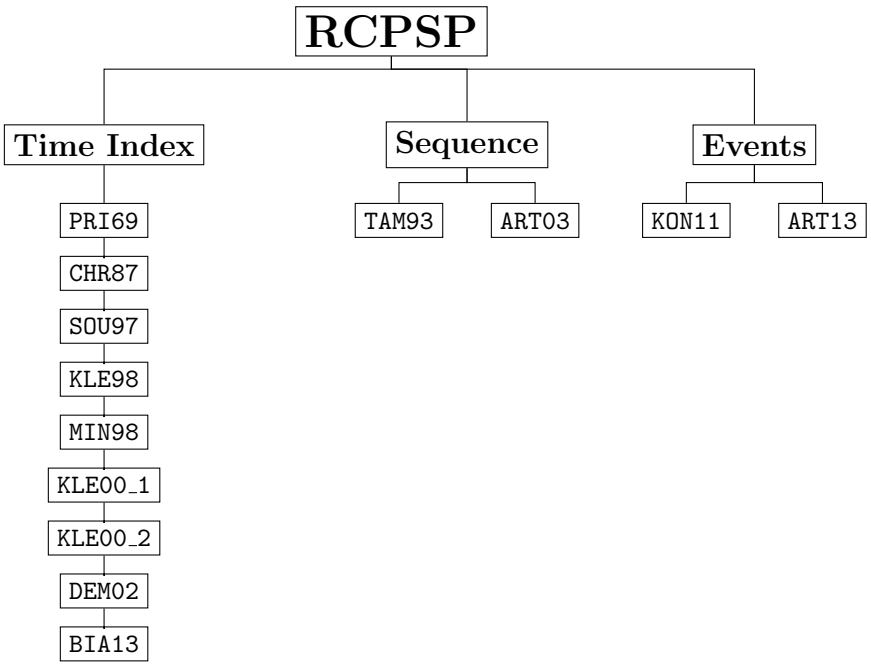


Fig. 4: RCPSP MILP formulations reported in the literature categorized by their family type.

In the context of employing a quantum annealer for solving the RCPSP, a pertinent question emerges: among the various available formulations, which one to choose? Specifically, which formulation aligns most effectively with the capabilities of current commercially accessible quantum annealers? Addressing these inquiries necessitates an investigation into the transformation of these formulations into their corresponding QUBO form. This evaluation involves assessing each formulation based on factors such as the resulting QUBO size, the need for additional slack supplementary variables, and the sparsity of the associated QUBO graph. These characteristics play a pivotal role in determining the suitability of each formulation for solving the RCPSP, particularly within the confines of NISQ-era quantum annealers. This study aims to provide comprehensive insights to answer these critical questions.

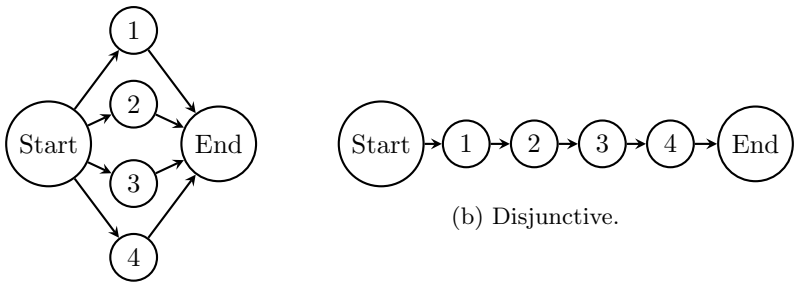
4 Methods

In the following section we explain different aspects of the experimental methodology followed in this study.

4.1 RCPSP instance selection protocol

A classical study for assessing the performance of solving the RCPSP would normally involve instances from the well-known PSPLIB dataset of Kolisch et al. [61, 62] or perhaps the more recent CV dataset of Coelho and Vanhoucke [63]. Both datasets offer a large number of hard instances of varying sizes (where size is defined by the number of project activities, ranging from 20 to 120 activities). However, considering the limitations on the number of available qubits in the D-wave Advantage 6.3 Quantum Annealer, we decided to utilize the RanGen instance generator proposed by Demeulemeester, Vanhoucke, and Herroelen [64, 65]. This generator allows the creation of random instances with varying levels of difficulty.

The instances used on this work were generated following the well-established protocol proposed by Baptiste and Le Pape [66], which involves constructing both disjunctive and cumulative instances. Disjunctive instances are characterized by a large number of precedence constraints, resulting in a highly sequential schedule with limited opportunities for parallel execution. On the other hand, cumulative instances exhibit fewer precedence constraints, providing ample opportunities for parallelism and posing a greater challenge for solving.



(a) Cumulative.

(b) Disjunctive.

Fig. 5: Baptiste and Le Pape instance types: a) the left graph corresponds to a cumulative RCPSP instance, b) right graph corresponds to a disjunctive RCPSP instance.

The RanGen generator facilitates the generation of both disjunctive and cumulative instances through a parameter called "Order Strength" (OS), ranging from 0 to 1. A value of 0 indicates a fully cumulative instance as in Fig. 5a, while 1 represents a fully disjunctive instance as in Fig. 5b. The generator also allows the tuning of resource constraints by specifying the number of resources, along with the level of constraint for renewable resources, this is done via a parameter called "Resource Constrainedness" (RC) that again ranges from 0 to 1 where 1 means that all activities consume the maximum resource capacity B_k, while 0 means that there is no resource consumption by the project activities.

In our study, we utilized RanGen to produce instances of the RCPSP across various dimensions, determined by the number of non-dummy activities (i.e., $n \in \{3, 4, 5, 6, 7, 8\}$). Within each size category, we generated three distinct instance types

characterized by OS values of 0.1, 0.5, and 0.9. We refer to these instance types as "Cumulative", "Medium OS", and "Disjunctive". Each instance features two renewable resources ($k = 2$) with a RC set to 0.5. The activity durations p_j, and resource consumption values b_{jk}, were constrained within the range of 1 to 2 units, to further limit the number of qubits required by the quantum annealer. Furthermore, for every instance, the maximum resource capacity B_k was established at 3 units.

4.2 Benchmarking metrics for quantum optimization

Benchmarking metrics play a pivotal role in the evaluation of adiabatic evolution computing algorithms and particularly for quantum annealing. In the context of benchmarking there are inherent challenges that arise from the unique nature of quantum annealers. First QA blends quantum and analog elements that are devoid of discrete instructions or basic operations that lend themselves to conventional counting methods used by classical computers. Consequently, resorting to runtime becomes a pragmatic yet intricate solution, given the transient and unstable nature of QA.

A further complication arises from the juxtaposition of hardware-implemented quantum-annealing algorithms against their software-implemented counterparts (simulated quantum annealing). Traditional benchmarks for computer platforms, software, and algorithms often fail to account for this mixed scenario, leading to a deficiency in standard guidelines for robust benchmarking.

Unlike classical benchmarks that often distill performance into a single metric, the evaluation of adiabatic evolution algorithms, particularly in the quantum realm, necessitates a more nuanced approach. The performance of a heuristic on a given input is aptly described by a curve delineating the trade-off between computation time and solution quality. This nuanced perspective requires a repertoire of metrics for comprehensive evaluation.

In the empirical evaluation of quantum annealing and adiabatic evolution algorithms, several performance metrics have surfaced in the literature. Notable among these are "Time-to-Solution" [67] (TTS), which centers on the total time required for a solver to identify a ground state (optimal solution) with a sufficiently high probability. One of the biggest disadvantages of TTS, is that it relies on a priori knowledge of the optimal solution, potentially overlooking benefits derived from near-ground state solutions.

Alternatively, "Time-to-Target" (TTT) [15] is a relevant metric that focuses on the overall time required by solvers to attain a target solution energy, determined by the energy distribution of the quantum-annealing processor. This metric provides a more versatile evaluation, acknowledging solutions that may not align with a predetermined optimal outcome.

Another recent interesting metric is the "Q-score" [68] conceived by the technology consulting company Atos. The Q-score measures the maximum number of qubits effectively employed to solve combinatorial optimization problem. Originally the Q-score was developed to evaluate the solution of the Max-Cut problem using the Quantum Approximate Optimization Algorithm [69] (QAOA), howver this metric can be readily adapted to assess the performance of quantum annealing as it was shown by Van der Schoot et al. [64]. The Q-score of a given problem can be then calculated by:

$$n^* \equiv \max\{n \in \mathbb{N}, \beta(n) > \beta^*\}. \quad (1)$$

With $\beta^* = 0.2$ (obtained empirically by studying the behavior of QAOA). and $\beta(n)$ determined by the following ratio:

$$\beta(n) = \frac{\tilde{C}(n) - C_r(n)}{C_{\max}(n) - C_r(n)}. \quad (2)$$

Where $\tilde{C}(n)$ is the average energy output obtained from QA, $C_r(n)$ is the average output value obtained from solving the QUBO/Ising problem with a random sampling method, and $C_{\max}(n)$ equal to the energy of the ground state.

For the specific focus of this paper, the metrics of interest include TTT and the Q-score. By concentrating on these metrics, we aim to ascertain the potential for speed-ups and determine the largest instance of the RCPSP, measured in terms of the number of project activities, that can be effectively addressed using the current Advantage 6.3 Quantum Annealing system from D-Wave.

4.3 QUBO penalty selection strategy

Adjusting the values of the multipliers weighing the penalties due to relaxed constraints is an important but difficult step. Given a QUBO problem where $E(x) = x^T Q x$, with a number of n binary variables $x_i, \forall i \in \mathcal{X}$ we can further decompose this problem into two portions $E(x) = v(x) + \lambda c(x)$ where $v(x)$ corresponds to the energy contribution to the objective function, $c(x)$ corresponds to the energy contribution due to the problem constraints, and λ represents the penalty weight. The main goal then is to find the value of λ such that the optimal solution to the penalised objective function is the optimal solution of the original constrained problem. Multiple strategies have been proposed in the literature, each offering a unique approach to determining the most effective penalty coefficients.

One of the earliest and simplest strategies was proposed by Lucas in 2014 [70]. This method involves utilizing the upper bound of the pure objective function, which is mathematically represented as $\lambda = x^T Q x$, where $x_i = 1, \forall i \in \mathcal{X}$ (i.e., the solution where all binary decision values are set to 1 is used as an upper bound of the energy minimization problem). This approach provides a straightforward and easily computable penalty.

Lucas also introduced another possibility for penalty selection, which involves using the maximum QUBO value, denoted as:

$$max(Q_{ij}) \quad \forall \quad (i,j) \in \mathcal{X}^2. \quad (3)$$

This method takes into account the highest interaction value between the variables, ensuring that the penalty is significant enough to enforce the constraints effectively.

More recently, Verma and Lewis in 2020 [71] proposed a more sophisticated method. Their approach estimates the potential gain or loss in the objective function value that could result from switching a particular bit on or off. This method provides a more nuanced and dynamic way of calculating penalties, potentially leading to more accurate and efficient solutions, especially in complex scenarios where the impact of each binary variable on the objective function is not uniform.

$$W_c = \left\{ -Q_{ii} - \sum_{\substack{j=1 \\ j \neq i}}^{n} \min\{Q_{ij}, 0\}, Q_{ii} + \sum_{\substack{j=1 \\ j \neq i}}^{n} \max\{Q_{ij}, 0\} \quad \forall i \in \mathcal{X} \right\}, \tag{4}$$

$$\lambda = \max_{i=1}^{n} W_{c_i}. \tag{5}$$

The array of strategies discussed underscores the dynamic and continuously evolving landscape of penalty selection in QUBO problems. Each approach, distinct in its methodology, brings its own set of strengths and is tailored to suit various problem types. As penalty selection stands as a vibrant and ongoing field of research, our study opts for a more foundational approach. We have chosen to employ a simple penalty selection method where the penalties are just a multiple of the sum of activity duration's p_i, $\lambda = \sum_{j=1}^{n} p_i$, leveraging the upper bound strategy, to ensure clarity and ease of implementation in our analysis.

4.4 Experimental setup

The experimental results presented in this study were derived from the utilization of the D-Wave Advantage 6.3 Quantum Annealer, featuring 5640 qubits and a connectivity of 15 connections per qubit. The Minor embeddings were established employing D-Wave's "minor-miner" heuristic. Each instance generated using the approach explained in Section 5.3 was solved on the QPU with a specific annealing time of $20\mu s$ (the decision to use $20\mu s$ is based on the analysis conducted in Section 5.4 and is also shown in Fig. 16) and, for every instance, 10000 samples were recorded. It is important to note that the mentioned annealing time exclusively pertains to quantum annealing and does not extend to Reverse Quantum Annealing (RQA), since RQA uses an specific annealing schedule described next.

Fig. 6 illustrates the implementation of RQA using a 4-point schedule: [[0.0, 1.0], [2.75, 0.45], [82.75, 0.45], [83.025, 1.0]]. The schedule involves a reverse evolution from $s = 1$ to $s = 0.45$ within the initial 2.75 µs, followed by an 80 µs pause. Subsequently, a forward evolution of 1 µs occurs, transitioning from $s = 0.45$ to $s = 1$. Here, the variable s represents the percentage of implementation of the problem Hamiltonian $\mathcal{H}_1$ during the annealing evolution.

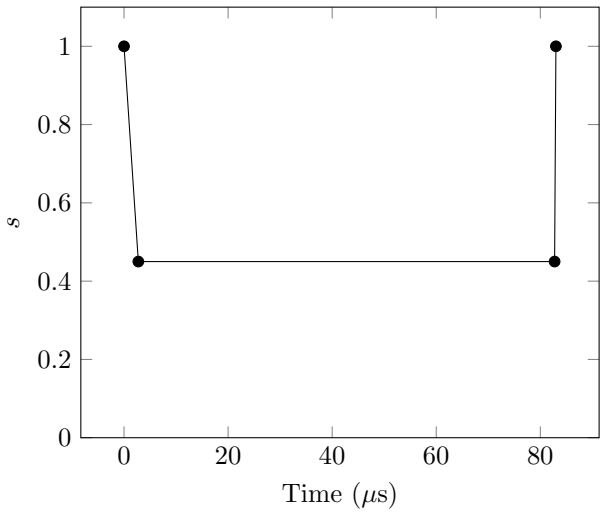


Fig. 6: Reverse quantum annealing schedule used for the RQA numerical results of this work.

The impact of including pauses in the annealing schedule on the performance of QA was assessed in Section 5.4. An annealing time of 20μs was employed, incorporating pauses of 2, 4, 6, 8, 12, 16, and 18μs, respectively. The resulting annealing schedules are visualized in Fig. 7. This analysis was comprehensive, covering all instance types and sizes. The evaluation involved recording the relative difference between the energy of the ground state E_0 and the minimum energy value achieved by QA, denoted as $E_{\min}$. This difference is calculated as $\left(\frac{E_{\min}-E_0}{E_0}\right)$. For each instance, 1000 samples were recorded. A similar analysis was conducted to evaluate the effect of different annealing times 1,5,10,20,50, and 100μs in the performance of QA.

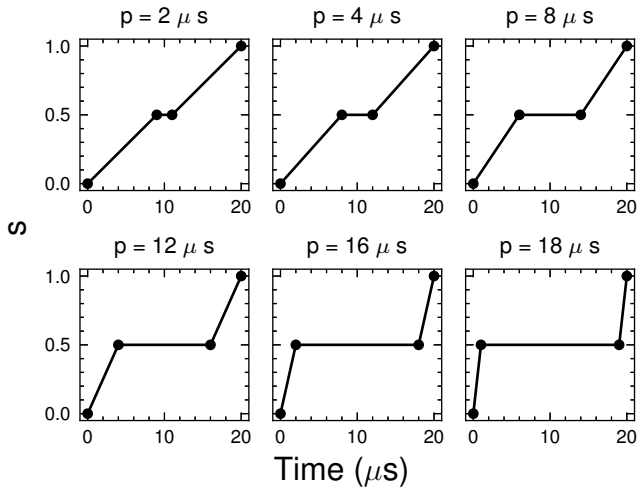


Fig. 7: Annealing schedules with pauses of different lengths (2,4,8,12,16, and 18μs) for a fixed annealing time of 20μs. These pauses correspond then to a 10,20,40,60,80, and 90% pause of the annealing time.

To create a comprehensive benchmark for assessing the performance of QA and RQA in this study, we have selected the following classical optimization techniques: Random Sampling (RS) and Simulated Annealing [18] (SA), along with classical solvers including GUROBI, COIN-CBC, and GLPK.

5 Results

5.1 QUBO analysis of MILP formulations for the RCPSP

In this section, we identify the optimal QUBO formulation for the RCPSP, focusing on the formulation that requires the fewest qubits. The minimal number of qubits is directly related to the number of slack variables, the type of variables, and the sparsity of the QUBO graph. Initially, we examine these crucial elements across 3 types of RCPSP formulations. Subsequently, from the eight formulations we have studied within the selected type, we determine the most effective one.

5.1.1 Analysis of formulation types

Once transformed into QUBO formulations, time-index, sequence, and event formulations (see Fig. 4) are compared with the data presented in Table C2. The number of qubits required for each formulation is reported in Fig. 8. This Fig. shows that the time-index formulation `PRI69` requires significantly fewer qubits compared to the sequence-based and event-based formulations. Moreover, the gap between these formulations increases in instance size. These results may appear counter intuitive when considering the original number of variables required by each formulation, as detailed in Table C2. Notably, sequence-based and event-based formulations exhibit a lower number of original variables than time-index formulations, this compactness characteristic has been well-documented by Koné [59].

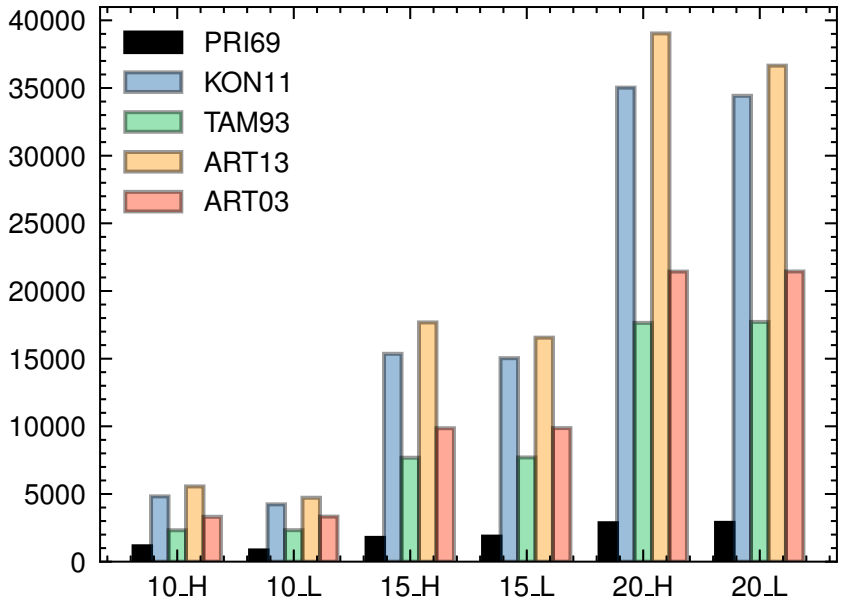


Fig. 8: Maximum number of qubits required by the different families of the RCPSP MILP formulations. H refers to disjunctive instances that have high OS (i.e., OS = 0.9) and L refers to cumulative instances that have low OS (i.e., OS = 0.1).

The apparent discrepancy in the number of QUBO variables can be elucidated by closely inspecting the nature of the different formulations. The `TAM93` formulation requires a priori knowledge of "forbidden sets" $\mathcal{F}$. These are sets of activities that share no precedence constraints and, when scheduled in parallel, violate resource constraints. Demeulemeester and Herroelen [72] demonstrated that generating the minimum forbidden sets entails a worst-case complexity of $O(2^n)$. Consequently, it logically demands more qubits compared to `PRI69`, especially as the instance size increases, since the size of the forbidden sets $\mathcal{F}$ increases accordingly.

In the `ART03` resource flow formulation, instead of using forbidden sets $\mathcal{F}$, integer variables ϕ_{ij}^k (where $i, j \in \mathcal{A}^2$ and $k \in R$) are employed, each requiring $\lfloor \log_2 \phi_{ij}^k \rfloor + 1$ binary qubits for transformation. These variables take part of multiple inequalities for maintaining resource constraints, leading to the need for additional slack integer variables, which also require binary transformation. Similarly, the event-based formulations `KON11` and `ART13` utilize integer variables t_e to represent event start times, with `KON11` also incorporating extra integers b_{ek} for resource consumption. These variables, also used in several constraints, require further slack variables for binary conversion.

5.1.2 Analysis of time-index formulations

Taking the same RCPSP instances as before, Table C3 presents the outcome resulting from the conversion of 8 distinct RCPSP time-index MILP formulations into QUBO.

The number of qubits per formulation is represented by Fig. 9 which shows the efficiency of the `PRI69` formulation compared to the other time-index formulations. This outcome is unsurprising given the inherent simplicity of `PRI69` that is characterized by fewer variables, a diminished count of inequalities, and consequently, a reduced number of slack variables. The next favorable option is the `MIN98` formulation; however, this choice entails the prerequisite knowledge of feasible sets $\mathcal{I}$. Similar to `TAM93`, the determination of these sets $\mathcal{I}$ entails a worst-case complexity of $O(2^n)$.

For the other formulations, the various strategies aimed at optimizing solutions for a classical computer prove to be costly in terms of the number of qubits required. We can highlight a few of them here: the `CHR87` formulation modifies the `PRI69` formulation by replacing its precedence constraint with additional inequalities. `SOU97` introduces "step" variables to signal the start of an activity, while `KLE98` considers a "step" formulation based on the MILP of a preemptive RCPSP [53]. The `KLE00_1` and `KLE00_2` formulations incorporate "on/off" variables. Additionally, `BIA13` integrates continuous binary variables indicating the percentage of activity completion while `MIN98`, bearing similarities to `TAM93`, requires prior knowledge of "feasible sets" $\mathcal{I}$.

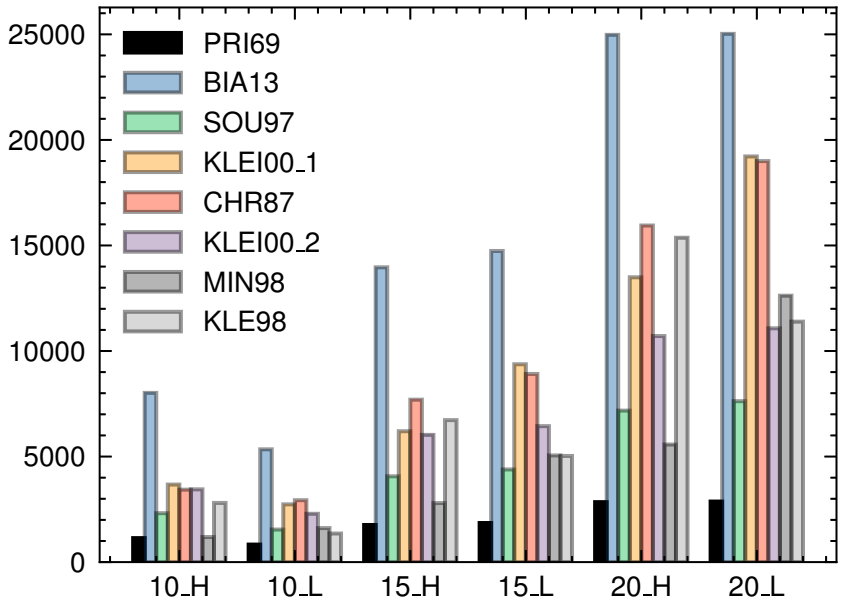


Fig. 9: Maximum Number of qubits required by the different time-index MILP formulations for the RCPSP. H refers to disjunctive instances that have high OS (i.e., OS = 0.9) and L refers to cumulative instances that have low OS (i.e., OS = 0.1).

5.2 RCPSP QUBO

Pritsker et al. [48] proposed `PRI69`, one of the first formulations for the RCPSP. Although it may initially seem counterintuitive, we have chosen this formulation as the basis for formulating our QUBO, based on the analysis in the previous section. We use the notations given in Table 2 to introduce the parameters.

Notation	Definition
n	Number of activities (excluding the two dummy activities indexed by 0 and $n+1$.)
$\mathcal{T}$	Maximum number of time periods.
$\mathcal{A}$	Set of Activities.
$\mathcal{H}$	$\mathcal{H} = 0, 1, ..., T$ is the scheduling horizon.
$\mathcal{E}$	Set of edges representing a precedence relationship between two activities. For instance, if i and j are two activities of $\mathcal{A}$, and if activity i must be finished to start activity j, $(i, j) \in \mathcal{E}$.
$\mathcal{R}$	Set of resources.
p_i	Processing time of the activity $i \in \mathcal{A}$.
B_k	Capacity of the resource $k, k \in \mathcal{R}$.
b_{ik}	Activity i consumption of the resource $k \in \mathcal{R}$ and its capacity B_k.

Table 2: RCPSP parameters used in the `PRI69` time-index formulation.

A single type of binary decision variable is considered, denoted as x_{it}, with $i \in \mathcal{A} \cup \{0, n+1\}$ and $t \in \mathcal{H}$. This variable is indexed by both the activities and the associated time. Each element x_{it}, $\forall i \in \mathcal{A} \cup \{0, n+1\}$ and $\forall t \in \mathcal{H}$, takes a $\{0; 1\}$ value such that:

$$x_{it} = \begin{cases} 1 \text{ if the activity } i \text{ starts at the period } t, \\ 0 \text{ otherwise.} \end{cases}$$

The initial objective function of [48] minimizes the starting time of the dummy-end activity. This function is noted $f(x)$ and its expression is given by (6). This is the basis of our QUBO which will then be completed with penalties corresponding to the relaxations of the constraints.

$$f(x) = \sum_t t x_{(n+1)t}. \tag{6}$$

Moreover we also consider the work of [9, 10] on the JSSP especially for the precedence constraints and the one-start constraint reformulation. For the latter, we force each activity to start exactly one time with the following set of constraints (7).

$$\sum_t x_{it} = 1, \qquad \forall\, i \in \mathcal{A}. \tag{7}$$

We relax these constraints for all activities and turn them into one penalty $P1$ given by the expression (8). Here, to ensure that the search for an optimal solution penalizes any infeasible solution, the expression is squared so that satisfying the constraint gives no penalty to the QUBO (i.e., remains equal to 0), and that violating the constraint increases the objective value (which we try to minimize).

$$P1(x) = \sum_{i=0}^{n+1} \left(\sum_{t} x_{it} - 1 \right)^2 . \tag{8}$$

We can model the precedence constraints of two consecutive activities by the inequalities (9).

$$\sum_{t \in \mathcal{H}} t x_{jt} \leq \sum_{t \in \mathcal{H}} t x_{it} + p_i \qquad \forall\ (i,j) \in \mathcal{E}. \tag{9}$$

The reformulation of all precedence constraints into a single penalty, denoted as $P2$, results in the expression shown in (10). Here, there is no need to square the decision variables, since any quadratic expression involving two binary variables never yields a negative value. It is easy to understand that the only scenario where such a constraint is not satisfied occurs when two consecutive activities erroneously start at the same time.

$$P2(x) = \sum_{(i,j) \in \mathcal{E}} \sum_{t \in \mathcal{H}} \sum_{\substack{t' \in \mathcal{H} \\ t' \setminus t + p_i > t'}} x_{it} x_{jt'} . \tag{10}$$

Whereas the JSSP has machine-sharing constraints which can also be modeled by simple quadratic expressions giving a penalty similar to $P2$, the RCPSP has another difficulty: the resource constraints. The inequalities (11) express these constraints as modeled in [48].

$$\sum_{i=1}^{n} b_{ik} \sum_{\tau = t - p_i + 1}^{t} x_{i\tau} \leq B_k \quad \forall t \in \mathcal{H}, \forall k \in R. \tag{11}$$

Since we need a penalty for relaxing the resource constraints, we add slack variables in order to reach an equality for each inequality of (11). Slack variables are noted $z_{tk}, t \in \mathcal{H}, k \in R$, as a non-negative integer for reformulating resource constraints (11) with the quadratic penalties (12).

$$P3(x) = \sum_{t} \sum_{k} \left(\sum_{i=1}^{n} b_{ik} \sum_{\tau = t - p_i + 1}^{t} x_{i\tau} - B_k + z_{tk} \right)^2 . \tag{12}$$

Since this reformulation aims to create a QUBO, the slack variables s_{tk} must correspond to binaries variables. Here, we can consider the minimum value of z_{tk} as zero and its maximum value as B_k. The related binary expression is given by equation (13) where y is a binary vector (each y^i takes value in {0,1}) and where the integer function $f(\alpha)$ gives the required maximum power of 2 with α the target integer.

$$z_{tk} = \sum_{i=0}^{f(B_k)} 2^i y^i \quad \forall t \in \mathcal{H}, \forall k \in \mathcal{R} \tag{13}$$

We note λ_1, λ_2, and λ_3 the multipliers balancing the penalties $P1$, $P2$, $P3$, respectively. The QUBO of the RCPSP can be formulated by the objective function

$f_{\text{QUBO}}(x)$ with the quadratic constraints (8), (10) and (12) such that $f_{\text{QUBO}}(x) = f(x) + \lambda_1 P1(x) + \lambda_2 P2(x) + \lambda_3 P3(x)$ as presented in equation (14).

$$\begin{aligned} f_{\text{QUBO}}(x) = & \sum_t t x_{(n+1)t} \\ & + \lambda_1 \sum_{i=0}^{n+1} (1 - \sum_t x_{it})^2 \\ & + \lambda_2 \sum_{(i,j)\in\mathcal{E}} \sum_t \sum_{t' \setminus t+p_i > t'} x_{it} x_{jt'} \\ & + \lambda_3 \sum_t \sum_k (\sum_{i=1}^{n} b_{ik} \sum_{\tau=t-p_i+1}^{t} x_{i\tau} - B_k + z_{tk})^2 \end{aligned} \tag{14}$$

5.3 Numerical results

Fig. 10 illustrates the TTT evolution for a range of optimization methods applied to a specific cumulative instance involving six non-dummy activities and two resources. Accompanying this, Fig. 11 presents the network diagram relevant to this instance. The figures are organized into a tripartite panel, demonstrating the progression of the optimization process across different energy quantiles. This progression is depicted from left to right, indicating a transition from higher to lower energy levels.

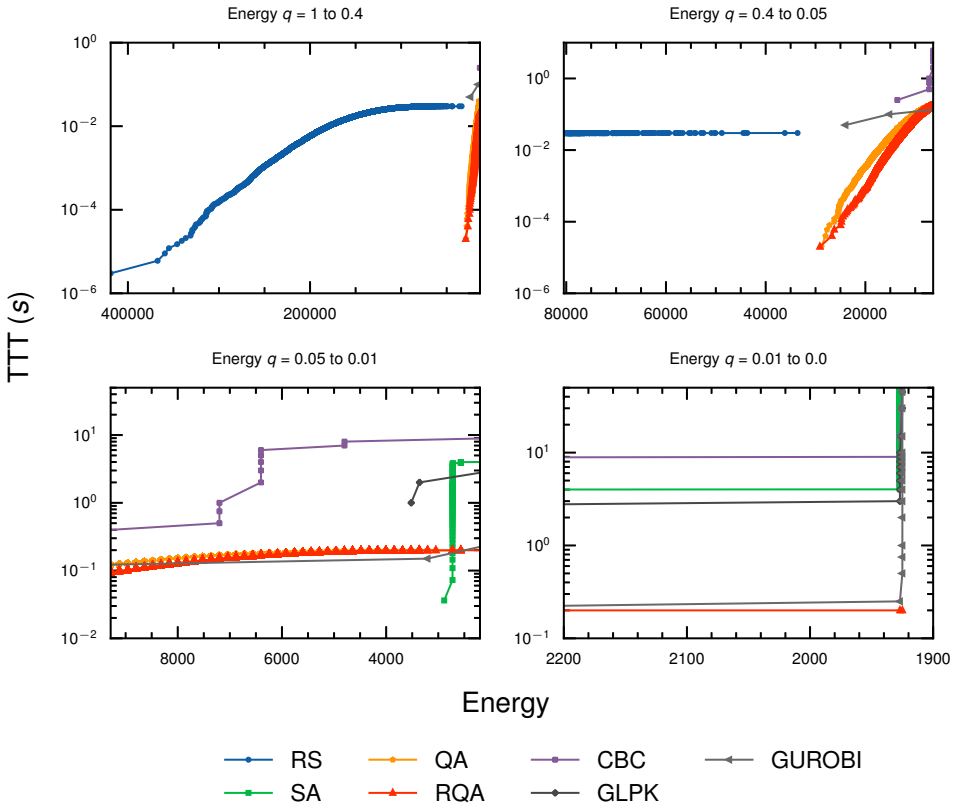


Fig. 10: TTT in seconds for a cumulative instance (OS = 0.1) of 6 non-dummy activities and 2 resources. The three panels show the evolution for different energy ranges, where energy decreases from left to right. The right most panel shows the energy range for the ground state of this instance.

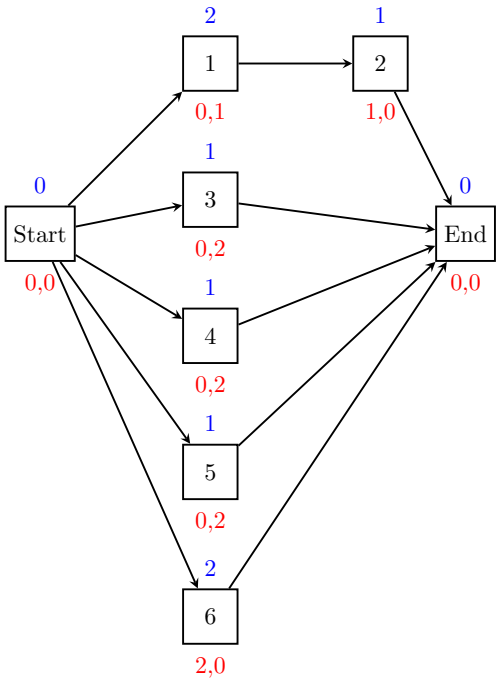
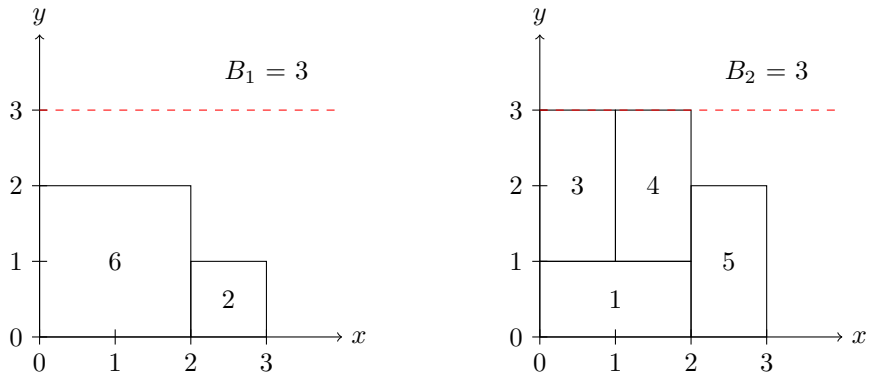


Fig. 11: Network diagram cumulative instance (OS = 0.1) with six non-dummy activities and two resources. The duration of each activity is displayed above each node whereas the resource consumption is displayed below.

It is noteworthy that QA and RQA exhibit discernible patterns, contrasting with the Random Sampling (RS) behavior observed in the evolution curve of RS. This observation suggests that QA and RQA are not mere random samplers.

In this specific case, RQA demonstrates exceptional performance, outperforming all other optimization methods, including the widely-used commercial solver GUROBI. This superior performance of RQA is particularly noticeable in the third panel of Fig. 10. Here, the distinctive red line marked with "△" symbols, representing RQA, reaches the ground state energy faster than any other method. Furthermore, Fig. 12 illustrates the optimal schedule derived using RQA. This schedule results in an optimal project make span of three days.



(a) Resource profile for resource 1.

(b) Resource profile for resource 2.

Fig. 12: Optimal schedule obtained from RQA for example cumulative instance (OS = 0.1) with six non-dummy activities and two resources.

Table 3 presents succinct results featuring the mean TTT values for solutions at different energy quantiles (0.9, 0.99, and 0.999). To provide further insight, Fig. 13 visually captures the distinctive mean TTT patterns, highlighting the performance variations among the evaluated optimization methods, for all instance types. Fig. D2, D3, and D4 from Appendix D offer further insight by illustrating the individual TTT performance for the different instance types evaluated in this work (cumulative, medium OS, and disjunctive).

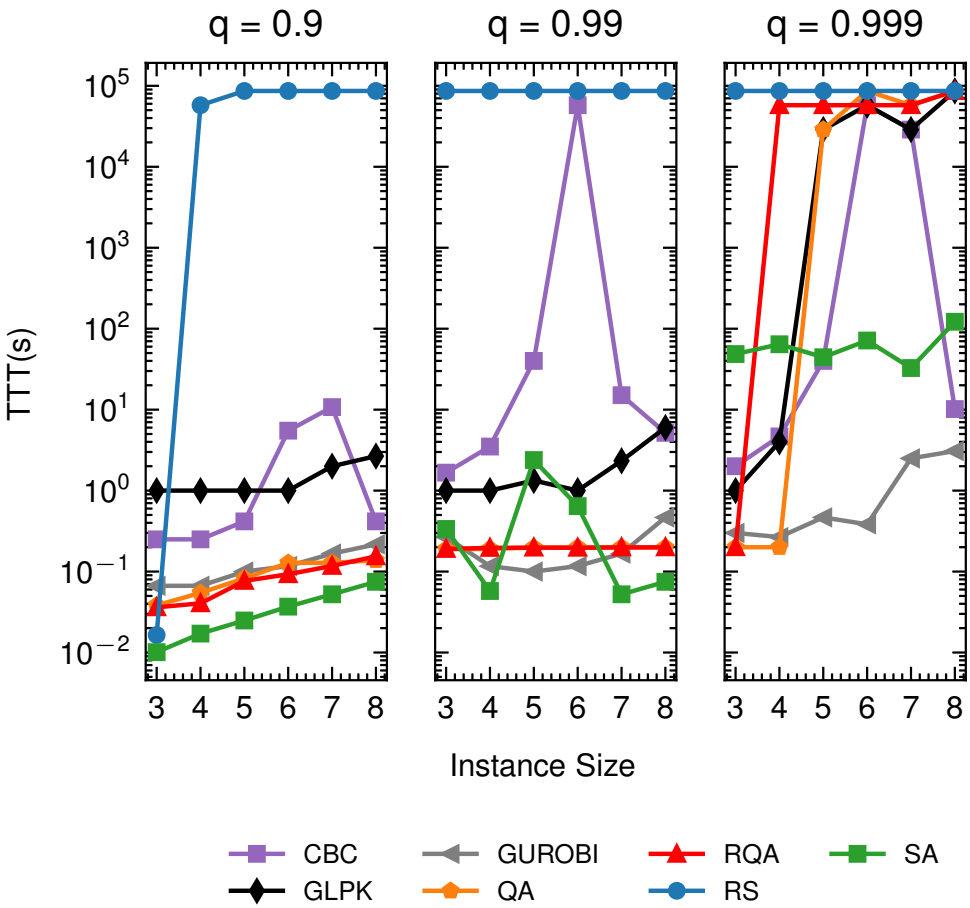


Fig. 13: Mean TTT, reported in seconds for all instance types; OS = 0.1, OS = 0.5, and OS= 0.9.

Size	Method	$q = 0.9$	$q = 0.95$	$q = 0.99$	$q = 0.999$	$q = 0.9999$
3	CBC	0.25	0.25	1.67	2	2
3	GLPK	1	1	1	1	1
3	GUROBI	0.07	0.07	0.27	0.3	0.3
3	QA	0.04	0.09	0.19	0.2	0.2
3	RQA	0.04	0.08	0.19	0.2	0.2
3	RS	0.02	57600	86400	86400	86400
3	SA	0.01	0.01	0.34	48.60	59.66
4	CBC	0.25	0.25	3.5	4.67	4.67
4	GLPK	1	1	1	4	28801.33
4	GUROBI	0.07	0.07	0.12	0.27	0.5
4	QA	0.05	0.12	0.19	0.20	28800.13
4	RQA	0.04	0.11	0.20	57600.07	86400
4	RS	57600.01	86400	86400	86400	86400
4	SA	0.02	0.02	0.06	64.20	93.92
5	CBC	0.42	0.67	40	40	45
5	GLPK	1	1	1.33	28802	57601
5	GUROBI	0.1	0.1	0.1	0.47	0.75
5	QA	0.08	0.16	0.20	28800.13	86400
5	RQA	0.08	0.16	0.20	57600.07	86400
5	RS	86400	86400	86400	86400	86400
5	SA	0.02	0.02	2.39	44.41	136.76
6	CBC	5.5	28803.17	57603	57603	57615
6	GLPK	1	1	1	57601	57610
6	GUROBI	0.12	0.12	0.12	0.38	2.17
6	QA	0.13	0.17	0.20	86400	86400
6	RQA	0.09	0.15	0.20	57600.07	86400
6	RS	86400	86400	86400	86400	86400
6	SA	0.04	0.04	0.64	71.46	159.07
7	CBC	10.75	15.08	15.08	28810.08	28810.08
7	GLPK	2	2	2.33	28820	86400
7	GUROBI	0.17	0.17	0.17	2.5	16.17
7	QA	0.13	0.18	0.20	57600.07	86400
7	RQA	0.12	0.17	0.20	57600.07	57600.07
7	RS	86400	86400	86400	86400	86400
7	SA	0.05	0.05	0.05	32.65	108.17
8	CBC	0.42	0.42	5.17	10.17	28800.17
8	GLPK	2.67	2.67	6	86400	86400
8	GUROBI	0.22	0.22	0.47	3.08	5
8	QA	0.14	0.18	0.20	86400	86400
8	RQA	0.15	0.18	0.20	86400	86400
8	RS	86400	86400	86400	86400	86400
8	SA	0.07	0.07	0.07	121.81	441.25

Table 3: TTT (s) mean results for Disjunctive, Cumulative, and Medium OS instances.

The results presented in Table 3 and Fig. D4, D2, and D3 reveal interesting dynamics. It is evident that QA and RQA face challenges in maintaining superiority, especially when tasked with finding ground states in instances of larger sizes. Despite this, in mere fractions of a second, both QA and, notably, RQA demonstrate

their capability to provide high-quality solutions, approaching the proximity of ground states.

Overall, for the high energy quantiles GUROBI exhibits consistent superiority over SA, QA, RQA, and other free solvers. However, this trend does not extend to certain freely available solvers, such as GLPK and CBC. Notably, in scenarios involving lower energy quantiles, both QA and RQA demonstrate significant advantages over these free solvers. This is particularly evident in the left ($q = 0.9$) and middle ($q = 0.99$) panels of Fig. 13, where QA and RQA outperform their counterparts in reaching lower energy states more efficiently.

Contrary to the findings reported by Carugno et al. [10] in the context of the JSSP, our results indicate that SA surpasses both QA and RQA in terms of performance across most instance sizes and energy quantiles. In agreement with their observations, however, RQA does tend to show superior performance compared to QA, particularly at lower energy quantiles.

Adapting the Q-score methodology for QA, as proposed by Van der Schoot et al. [64], we computed the $\beta(n)$ values for various instance sizes and types examined in our study. Analysis of Fig. 14 reveals that the $\beta(n)$ values for both QA and RQA consistently exceed the critical threshold of 0.2. However, it's important to remind the reader that this threshold was empirically established based on the solution of instances of the the Max-Cut problem using QAOA. Therefore, its validity is based solely on these empirical findings, without additional theoretical support.

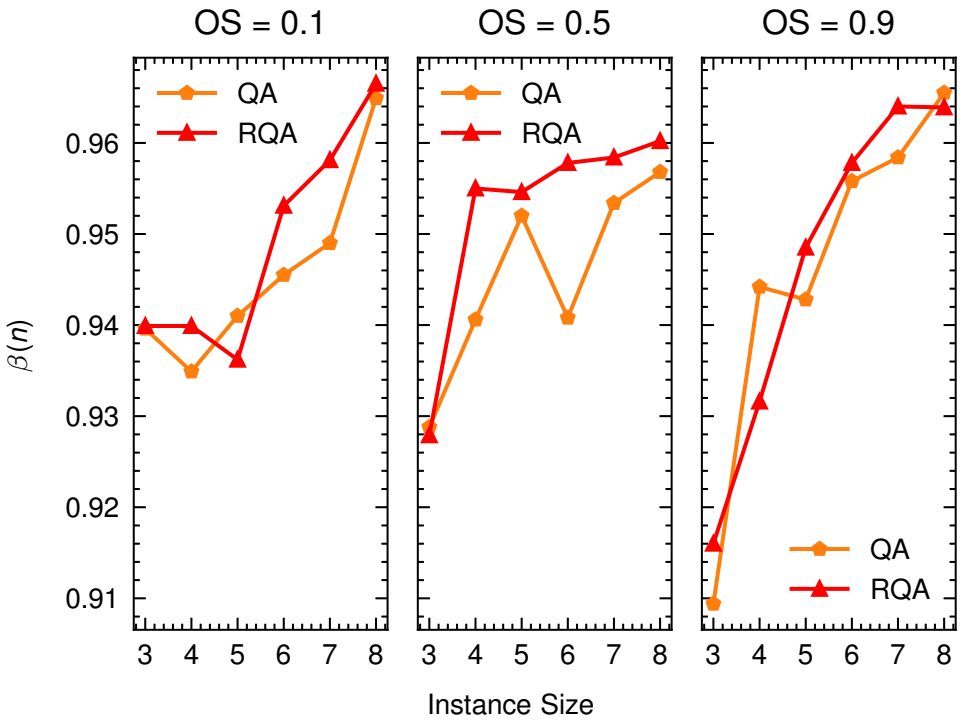


Fig. 14: $\beta(n)$ for all instances with 3, 4, 5, 6, 7, and 8 non-dummy activities for all instance types (Cumulative, Medium OS, and Disjunctive).

We therefore conclude that the estimated maximum instance size of the RCPSP that can be solved using a D-Wave quantum annealer peaks at seven non-dummy activities. Although the $\beta(n)$ values substantially exceed the critical threshold, we observe a rapid decline in solution quality beyond this activity count. This deterioration is evident when examining the mean "chain-break" percentage reported by the D-Wave system. For instances with eight activities or more, the average chain-break percentage escalates to approximately 74%, indicating a significant breakdown among the logical qubits that were utilized to map the original QUBO graph onto the QPU topology. With such a high percentage of chain breaks, the reliability of the annealer in solving the intended optimization problem becomes questionable, despite the higher $\beta(n)$ values observed.

5.4 Anneal time and pausing effects

In the preceding section, our experimental approach was predicated on determining an optimal annealing time for QA. This determination was made empirically, analyzing the performance of QA across a spectrum of annealing times: 1, 5, 10, 20, 50, and 100μs.

The performance evaluation of QA for each annealing time was determined based on a relative energy difference metric. This metric juxtaposes the minimum energy achieved through QA against the ground state energy for the given instances as $\left(\frac{E_{\min}-E_0}{E_0}\right)$. Our analysis spanned instance with 3, 4, 5, 6, 7, and 8 non-dummy activities and all instance types (Cumulative, Medium OS, and Disjunctive). For each configuration we collected 1000 samples.

Insights into the mean behavior of QA across these different annealing times are presented in Fig. 15. This line graph illustrates the relationship between the instance size (x-axis) and the relative energy differences (y-axis), offering a visual representation of the performance variations across instance sizes.

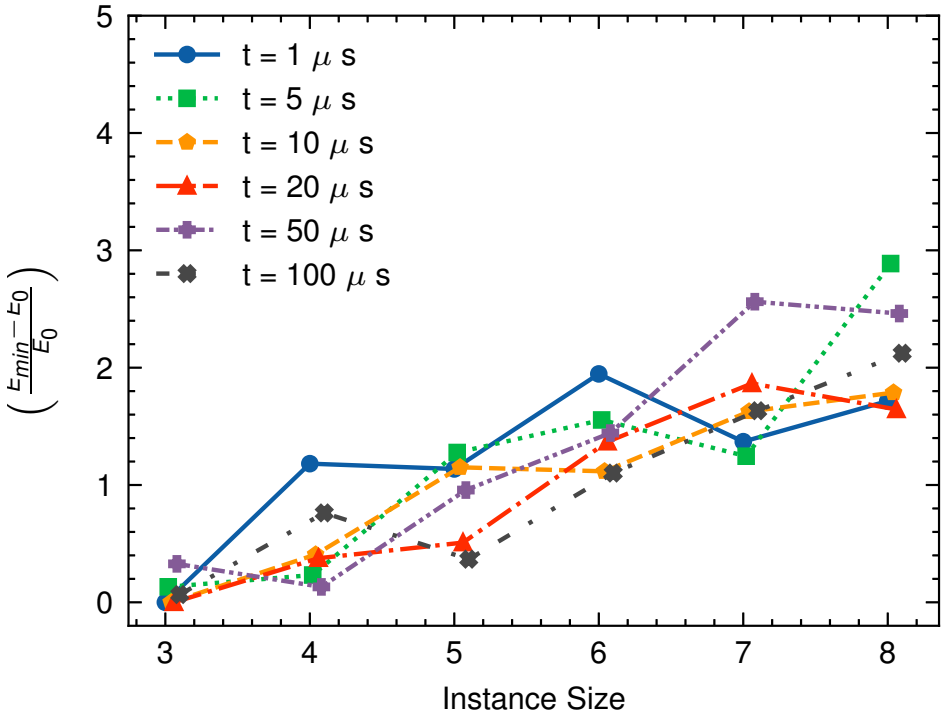


Fig. 15: Mean effect of annealing time on QA performance, measured as the mean relative deviation from the ground state $\left(\frac{E_{\min}-E_0}{E_0}\right)$ for all instance types (Cumulative, Medium OS, and Disjunctive).

Complementing this analysis, Fig. 16 presents a holistic view of the mean behavior across all instance sizes and types. Here, the annealing times are mapped along the x-axis, and the corresponding relative energy differences are charted on the y-axis. In accordance with [15], this graph reveals that an annealing time of approximately

$20\mu s$ emerges as optimal. Beyond this threshold, there is a discernible decline in QA performance.

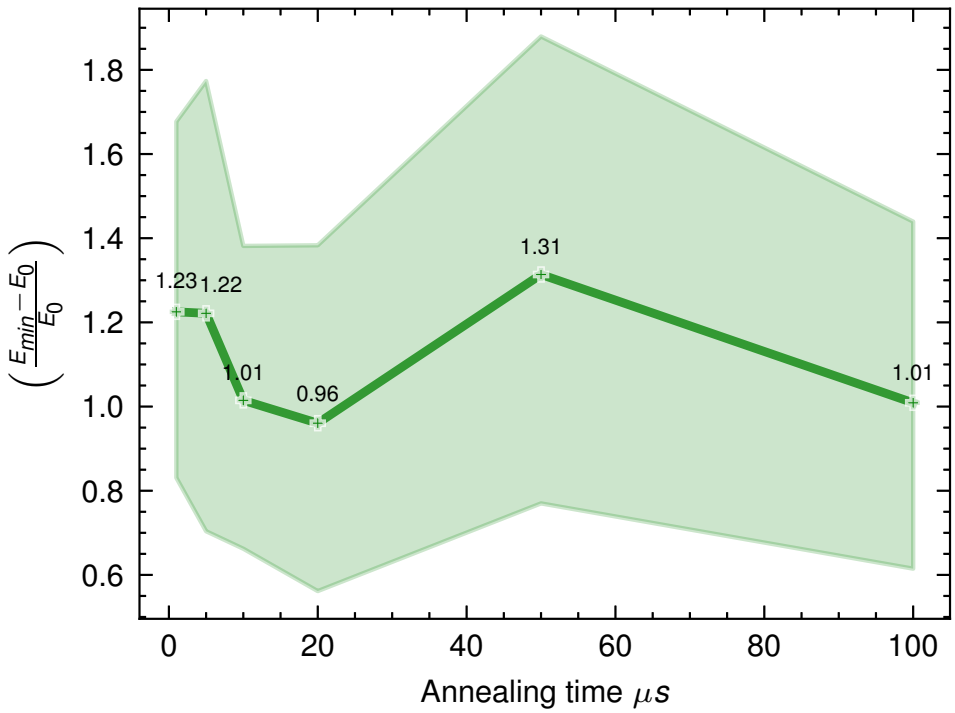


Fig. 16: The figure illustrates the impact of annealing times on QA performance, depicted as a function of the mean relative deviation from the ground state $\left(\frac{E_{\min}-E_0}{E_0}\right)$ on the y-axis, across various instance sizes and types. This relationship is represented by a green line, accompanied by its standard deviation delineated as a light green contour. The x-axis quantifies the annealing time.

Given these findings, we chose to conduct our experiments with an annealing time fixed at $20\mu s$. This decision was guided by the empirical evidence suggesting that this duration strikes an effective balance in optimizing QA performance. Fig. E5, E6, and E7 show the effect of annealing times for Cumulative, Medium OS, and Disjunctive instance types.

In continuation of our exploration into the multiple parameters involved in QA, we extended the investigation to assess the impact of annealing pauses on QA performance. This analysis aligns with the relative energy difference framework established earlier, comparing the minimum energy achieved via QA to the ground state energy for a particular instance.

Our focus centered on a fixed annealing time of $20\mu s$, examining the effects of pauses at intervals of 2, 4, 8, 12, 16, and $18\mu s$ (refer to Fig. 7 for a visual represenation of the annealing schedules, incorporating these pauses).

This investigation was inspired by the findings of Marshal et al. [73], who report performance enhancements in QA with the inclusion of annealing pauses. Our results, illustrated in Fig. 17, confirm this observation. Fig. 17 presents a heat map that encapsulates the QA performance across various instance sizes (3, 4, 5, 6, 7, and 8 non-dummy activities) and pause durations (shown on the y-axis as the percentage of pause relative to the total annealing time). Each cell in this heat map represents the average relative energy value derived from 1000 samples across all instance types (Cumulative, Medium OS, and Disjunctive), providing a comprehensive view of the performance landscape.

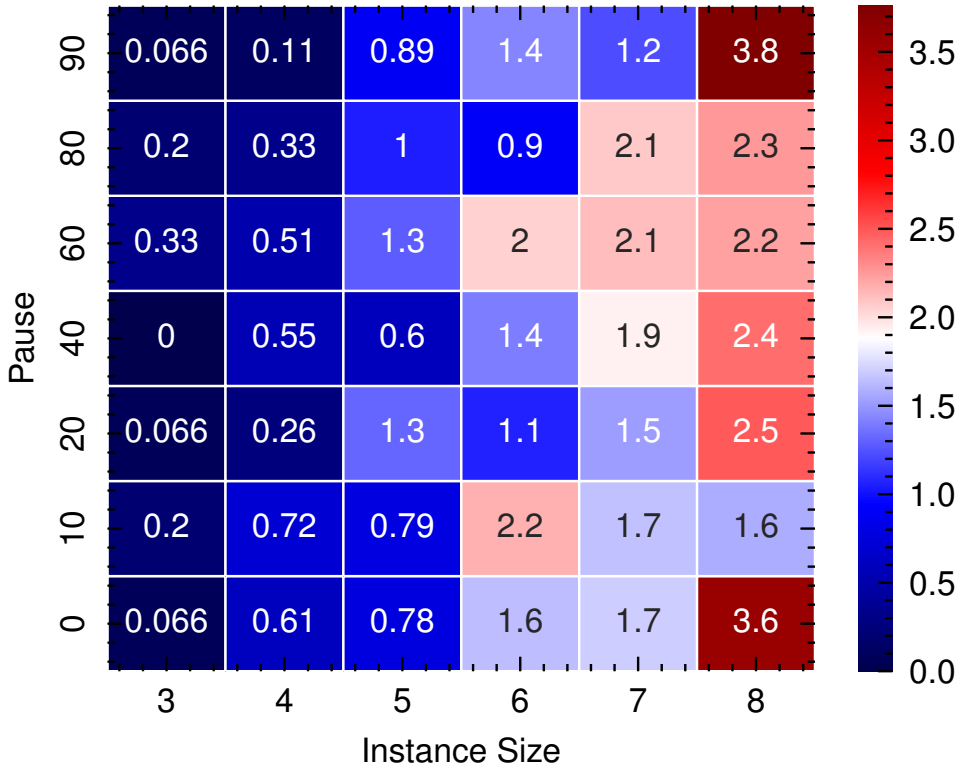


Fig. 17: Heatmap of the effect of pauses included in the annealing schedule vs the relative deviation from the ground state energy $\left(\frac{E_{\min}-E_0}{E_0}\right)$ for all instance types. The y-axis shows the percentage of pause in the annealing schedule, as shown in Fig. 7.

Intriguingly, Fig. 18 highlights that annealing pauses constituting 20% to 40% of the overall schedule (equating to $4\mu s$ and $8\mu s$ in this context) yield the most favorable

performance. Fig. F8, F9, and F10 show the effect of annealing pauses for Cumulative, Medium OS, and Disjunctive instance types.

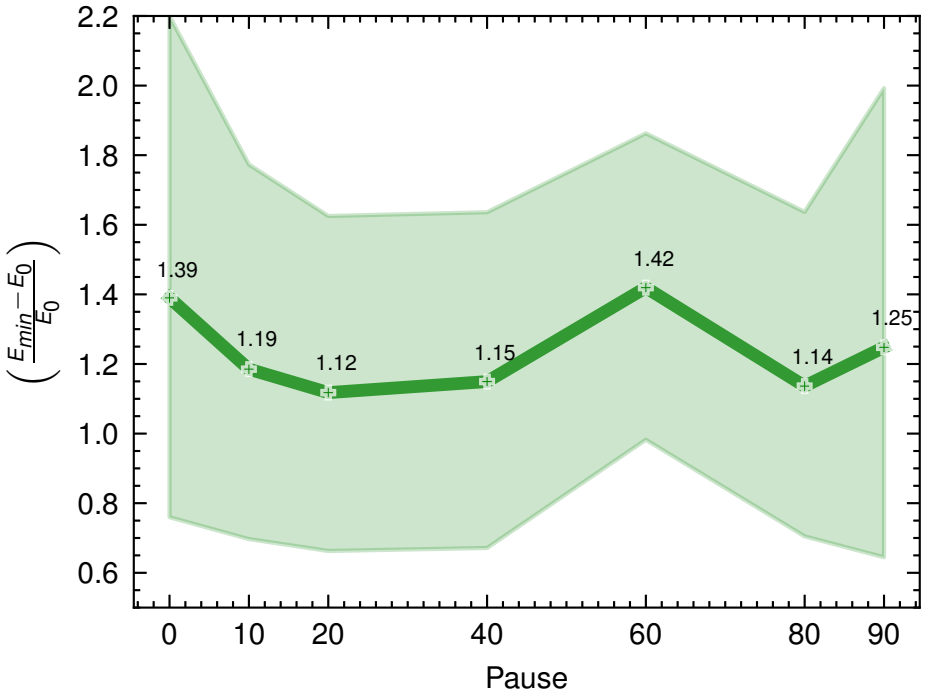


Fig. 18: General effect of pauses included in the annealing schedule vs the relative deviation from the ground state energy $\left(\frac{E_{\min}-E_0}{E_0}\right)$ for all instances. The y-axis shows the percentage of pause in the annealing schedule, as shown in Fig. 7.

It is important to note, however, that the inherent nature of RQA always incorporates a pause in the annealing schedule. Consequently, the insights obtained from this analysis predominantly apply to conventional QA rather than RQA.

6 Discussion

In this work, we have conducted a comprehensive exploration into the practical implementation of the RCPSP using quantum annealing, specifically by leveraging D-Wave's quantum-computing technology. Our investigation encompasses a range of aspects, from the basics of quantum annealing to advanced techniques like reverse quantum annealing.

To the best of our knowledge, we are the first to address the RCPSP using a quantum annealer, marking a significant contribution to the integration of quantum

computing in the field of operations research. Our work underscores the importance of QUBO modeling for solving RCPSP instances on quantum-annealing machines. We have conducted a thorough analysis of twelve well-known MILP formulations for RCPSP and convert them into a QUBO format. This includes identifying the most suitable formulation for quantum annealing, specifically the `PRI69` formulation, and providing the corresponding QUBO model.

It is worthy to note that the QUBO model (14) derived in Section 5.2 can easily be adapted to other variants of the RCPSP as in the case of the 1-preemptive resource-constrained project scheduling problem (1_PRCPSP). Adapting the formulation proposed by Ballestín et al. [74], we obtain the QUBO $f(x)^{\text{1\_PRCPSP}}_{\text{QUBO}}$ (presented in Appendix G1). The problem does not suffer from the addition of new slack variables. Indeed, the new constraints can be modeled as a quadratic product that must remain zero to be satisfied, similar to precedence constraints.

Other versions of the RCPSP can be based on the QUBO introduced in this study without the need to add many constraints. For example, if resources have periods of unavailability, or if their capacity is reduced (B_k becomes B_{kt}), it is sufficient to treat these periods as distinct activities with fixed variable values as it was shown by Hartmann [75]. This approach can also be applied when the use of resources need to be stopped (e.g., for maintenance operations) without a pre-determined period. In such cases, these periods of unavailability should be considered as activities, but without setting fixed variables. In these scenarios, it's important to establish a precedence constraint for each virtual activity to ensure that the identification of the unavailability period is properly included in the optimization process. However, this modification will require the inclusion of additional slack variables associated with the resource constraint.

In this work, we solve three distinct categories of problem instances, as defined by the protocol of Baptiste and Le Pape's, using RCPSP QUBO formulation (14) on a quantum annealer. Strategies such as developing a reverse quantum annealing schedule are employed. The results are then compared with those obtained from a classical solver. For this comparison, we conduct a detailed exploration of multiple metrics available for evaluating the performance of quantum annealing. Ultimately, we adopt the TTT and the Q-score for our evaluation. Additionally, we determine the maximum instance size that can still be solved with a machine equipped with 5000 qubits. Another key discovery from our research underscores the potential benefits of quantum annealing, especially in scenarios with constrained time limits, such as those encountered in online operations research problems. In addition to our primary research focus, we conduct an extensive and meticulous analysis to understand the impact of annealing times and the role of pauses within annealing schedules on the performance of QA.

While our research offers valuable insights, it's important to recognize its limitations. These include using problem instances of limited size, excluding other promising quantum-computing techniques like QAOA [69] due to hardware constraints, and the dependence on the "minor-miner" heuristic for the embedding procedure.

In alignment with the work of Venturelli et al. [9] and Carugno et al [10] on the JSSP, we recognize the potential for significant reduction in time horizons for the

RCPSP. This reduction can be achieved by pre-establishing upper bounds, derived from heuristics that can be calculated in polynomial time [76]. Such bounds can considerably reduce the number of variables required by time-index formulations like "PRI69". However, in this study, our focus was centered on examining the intrinsic performance of QA and RQA without incorporating these heuristic techniques.

These aspects warrant consideration in future research. Additionally, investigating alternative quantum-computing technologies, such as neutral atoms technology by Pasqal [77], is promising and deserves further exploration.

In conclusion, despite the acknowledged limitations, this work serves as a pioneering effort in applying quantum annealing to the RCPSP. We hope that our findings and methodologies will act as a catalyst for future benchmarks, fostering advancements towards practical applications using quantum computers in operations research.

Appendix A UML activity diagram: resolution process with a quantum-annealing solver

The Unified Modeling Language (UML) activity diagram presented in Fig. A1 summarizes the process of solving an optimization problem such as the RCPSP by implementing QA on a quantum machine. The activities are divided into two sets: on the left, human activities corresponding to problem modeling, and on the right, those automatically executed by the machine. The machine takes as input either an Ising model or a QUBO model. If the user only has a MILP, as is the case in this work for 12 different formulations, they must formulate it either as an Ising model or a QUBO model. If the latter is provided, the machine automatically converts it to Ising, as the two models are isomorphic. In addition, the user also provides weights corresponding to the penalties of each series of relaxed constraints (i.e., the multipliers λ_i for each constraint i).

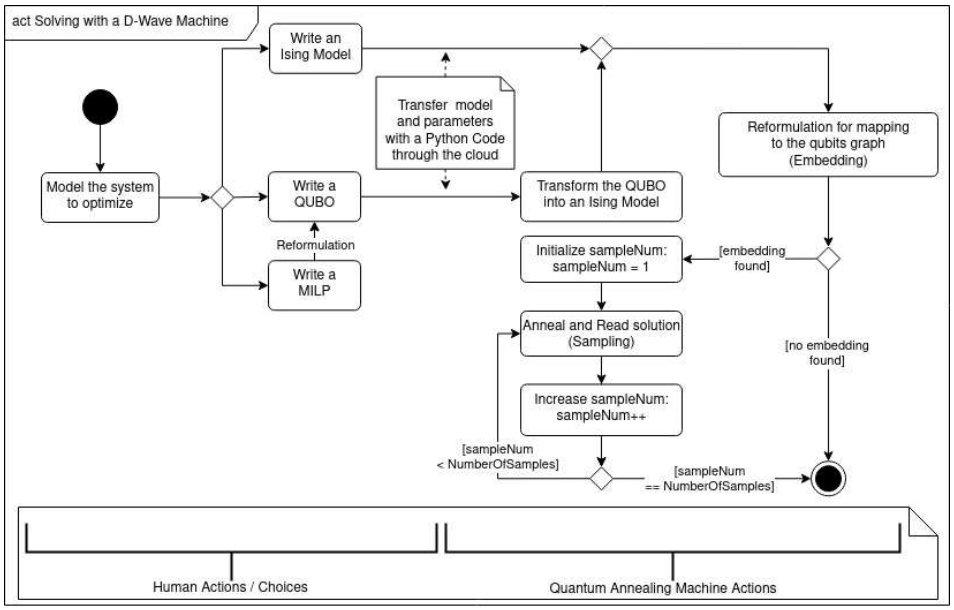


Fig. A1: UML activity diagram: transitioning from the mathematical model to the quantum annealing solution process. Each activity belongs either to the actions to be performed by the user or to the actions to be executed by the machine. The user provides a QUBO or an ISING model to the machine, and it replicates the QA as many times as requested by the user, resulting in as many solutions obtained at the end of the process.

Once the machine has an Ising model to process, the corresponding non-directed graph must be adapted to the machine's topology, that is, the qubits network. Thus,

the Ising model is transformed into another non-directed graph suited to the topology. The resulting graph is generally less dense to match the machine's connectivity, using more qubits for this purpose. This process, called "embedding", solves another optimization problem that the machine resolves heuristically. At the end of this process, if the machine does not find an "embedding", the experiment stops. Otherwise, the QA process is applied to the mapped graph.

The number of QA replications is determined by the number of *anneals* and *reads*, collectively referred to as *samples*. Users must specify two key parameters for each QA experiment conducted on a D-Wave machine: the number of *samples* and the time available for each anneal. These parameters are crucial and significantly influence the outcome of the experiment. Given that each experiment is typically restricted to a duration of about one second, the total time available for quantum machine access is primarily allocated between these two parameters. The optimal balance between them is usually determined empirically and varies depending on the specific optimization problem being addressed.

As output, the user obtains as many solutions as there were anneals executed. Even if all obtained solutions are feasible in the sense that the variable type is respected (binary) and no constraint violates the QUBO and Ising models, the feasibility of the solutions must be verified through the constraints of the initial MILP model, an operation that can be done in polynomial time.

Appendix B List of Symbols

Symbol	Definition
$\mathcal{H}(t)$	Time-dependent energy Hamiltonian
$\mathcal{H}_0$	Tunneling Hamiltonian
$\mathcal{H}_1$	Problem Hamiltonian
$A(t)$	Annealing schedule function associated with $\mathcal{H}_0$
$B(t)$	Annealing schedule function associated with $\mathcal{H}_1$
$\mathcal{P}$	Optimization Problem
G	Network graph of $\mathcal{P}$
V	Nodes of G
E	Edges of G
s_j	Spin variable $\in \{-1, 1\}$, associated with qubit j
h_j	Linear term of the Ising model, associated with spin variable s_j
$J_{i,j}$	Interaction term of the Ising model, associated with spin variables s_i and s_j
x_j	QUBO Binary variable $\in \{0, 1\}$ associated with qubit j
n	Number of project activities
$\mathcal{A}$	Set of project activities
$\mathcal{E}$	Set of precedence constraints
p_j	Duration of project activity j
$b_{j,k}$	Resource consumption of activity j for resource k
$\mathcal{R}$	Set of project renewable resources
$\mathcal{S}$	Project solution schedule
S_j	Start date of activity j in Schedule $\mathcal{S}$
B_k	Maximum capacity of resource k
$\mathcal{H}$	Set of available time-slots t
$\mathcal{T}$	Planning horizon (max value of t
$\beta(n)$	Q-Score ratio
β^*	Q-score critical ratio threshold
$\tilde{C}(n)$	Average energy value obtained from solutions of QA for problem $\mathcal{P}$
$C_{max}(n)$	minimum solution value of problem $\mathcal{P}$
$C_r(n)$	Average energy value obtained from solutions of RS for problem $\mathcal{P}$
n^*	Max number of qubits for which $\beta(n) \geq \beta^*$
Q	QUBO Matrix
$Q_{i,j}$	QUBO Interaction coefficient of variables i and j in matrix Q
$\mathcal{X}$	set of binary QUBO variables
x	QUBO solution
$E(x)$	Total QUBO energy of solution x
$v(x)$	QUBO objective function energy of solution x
$c(x)$	QUBO energy contribution of constraints for solution x
λ	QUBO Penalty term
W_c	Verma and Lewis penalty function
s	Standardized forward annealing schedule progress. 0 is no progress and 1 is 100% progress.
E_{min}	Minimum energy obtained from solution method when solving problem $\mathcal{P}$
E_0	Ground state energy of problem $\mathcal{P}$
$\mathcal{F}$	Forbidden sets of `TAM93`
$\mathcal{I}$	Feasible sets of `MIN98`
$\phi_{i,j}^k$	`ART03` Resource flow between activities i and j for resource k
t	time (integer units)
t_e	scheduled start-time for event e (Formulations of `KON11` and `ART13`)
$z_{t,k}$	Slack variables used for resource constraints of RCPSP QUBO

Table B1: List of Symbols

Appendix C Evaluation of MILP formulations for RCPSP

Table C2: Evaluation of MILP formulations for RCPSP.

Instance	Type	Family	MILP	n MILP	n (QUBO)	S (QUBO)	ρ
10_H	Disjunctive	Time Index	PRI69	570	1136	566	0.1566
10_H	Disjunctive	Sequence	TAM93	110	2321	2161	0.0073
10_H	Disjunctive	Sequence	ART03	310	3316	2356	0.0094
10_H	Disjunctive	Event	ART13	230	5559	5219	0.0056
10_H	Disjunctive	Event	KON11	111	4824	4658	0.0070
10_L	Cumulative	Time Index	PRI69	380	830	450	0.1740
10_L	Cumulative	Sequence	TAM93	110	2328	2168	0.0073
10_L	Cumulative	Sequence	ART03	310	3329	2369	0.0093
10_L	Cumulative	Event	ART13	230	4724	4384	0.0068
10_L	Cumulative	Event	KON11	111	4235	4069	0.0080
15_H	Disjunctive	Time Index	PRI69	990	1758	768	0.1443
15_H	Disjunctive	Sequence	TAM93	240	7681	7351	0.0022
15_H	Disjunctive	Sequence	ART03	690	9859	7729	0.0035
15_H	Disjunctive	Event	ART13	495	17678	17003	0.0021
15_H	Disjunctive	Event	KON11	241	15360	15023	0.0026
15_L	Cumulative	Time Index	PRI69	1035	1851	816	0.1526
15_L	Cumulative	Sequence	TAM93	240	7699	7369	0.0022
15_L	Cumulative	Sequence	ART03	690	9869	7739	0.0035
15_L	Cumulative	Event	ART13	495	16553	15878	0.0023
15_L	Cumulative	Event	KON11	241	15036	14699	0.0027
20_H	Disjunctive	Time Index	PRI69	1780	2840	1060	0.1318
20_H	Disjunctive	Sequence	TAM93	420	17645	17105	0.0009
20_H	Disjunctive	Sequence	ART03	1220	21435	17695	0.0017
20_H	Disjunctive	Event	ART13	860	39033	37933	0.0010
20_H	Disjunctive	Event	KON11	421	35028	34481	0.0012
20_L	Cumulative	Time Index	PRI69	1740	2868	1128	0.1427
20_L	Cumulative	Sequence	TAM93	420	17715	17175	0.0009
20_L	Cumulative	Sequence	ART03	1220	21443	17703	0.0017
20_L	Cumulative	Event	ART13	860	36653	35553	0.0011
20_L	Cumulative	Event	KON11	421	34426	33879	0.0012

n MILP = number of variables original problem, n (QUBO) = number of qubits, S (QUBO) = number of slack qubits, ρ = density of the QUBO

Table C3: Evaluation of MILP formulations for RCPSP.

Instance	Type	MILP	n MILP	n (QUBO)	S (QUBO)	ρ
10_H	Disjunctive	PRI69	570	1136	566	0.157
10_H	Disjunctive	CHR87	570	3432	2862	0.041
10_H	Disjunctive	SOU97	570	2316	1746	0.009
10_H	Disjunctive	KLE98	570	2803	2290	0.009
10_H	Disjunctive	KLE00_1	570	3673	3103	0.010
10_H	Disjunctive	KLE00_2	1140	3447	2307	0.006
10_H	Disjunctive	BIA13	1710	8016	2886	0.016
10_H	Disjunctive	MIN98	684	1187	503	0.099
10_L	Cumulative	PRI69	380	830	450	0.174
10_L	Cumulative	CHR87	380	2932	2552	0.034
10_L	Cumulative	SOU97	380	1542	1162	0.010
10_L	Cumulative	KLE98	380	1354	1012	0.023
10_L	Cumulative	KLE00_1	380	2735	2355	0.014
10_L	Cumulative	KLE00_2	760	2288	1528	0.013
10_L	Cumulative	BIA13	1140	5342	1922	0.023
10_L	Cumulative	MIN98	646	1611	965	0.065
15_H	Disjunctive	PRI69	990	1758	768	0.1443
15_H	Disjunctive	CHR87	990	7694	6704	0.022
15_H	Disjunctive	SOU97	990	4065	3075	0.006
15_H	Disjunctive	KLE98	990	6726	5802	0.009
15_H	Disjunctive	KLE00_1	990	6199	5209	0.008
15_H	Disjunctive	KLE00_2	1980	6027	4047	0.004
15_H	Disjunctive	BIA13	2970	13965	5055	0.013
15_H	Disjunctive	MIN98	1320	2796	1476	0.046
15_L	Cumulative	PRI69	1035	1851	816	0.153
15_L	Cumulative	CHR87	1035	8913	7878	0.020
15_L	Cumulative	SOU97	1035	4389	3354	0.006
15_L	Cumulative	KLE98	1035	5029	4063	0.011
15_L	Cumulative	KLE00_1	1035	9372	8337	0.007
15_L	Cumulative	KLE00_2	2070	6441	4371	0.008
15_L	Cumulative	BIA13	3105	14739	5424	0.012
15_L	Cumulative	MIN98	2001	5054	3053	0.055
20_H	Disjunctive	PRI69	1780	2840	1060	0.132
20_H	Disjunctive	CHR87	1780	15945	14165	0.014
20_H	Disjunctive	SOU97	1780	7177	5397	0.004
20_H	Disjunctive	KLE98	1780	15366	13675	0.005
20_H	Disjunctive	KLE00_1	1780	13492	11712	0.005
20_H	Disjunctive	KLE00_2	3560	10709	7149	0.003
20_H	Disjunctive	BIA13	5340	24977	8957	0.010
20_H	Disjunctive	MIN98	2581	5570	2989	0.030
20_L	Cumulative	PRI69	1740	2868	1128	0.143
20_L	Cumulative	CHR87	1740	18997	17257	0.012
20_L	Cumulative	SOU97	1740	7624	5884	0.004
20_L	Cumulative	KLE98	1740	11391	9738	0.007
20_L	Cumulative	KLE00_1	1740	19217	17477	0.004
20_L	Cumulative	KLE00_2	3480	11076	7596	0.006
20_L	Cumulative	BIA13	5220	25024	9364	0.009
20_L	Cumulative	MIN98	5307	12620	7313	0.089

n MILP = number of variables original problem, n (QUBO) = number of qubits, S (QUBO) = number of slack qubits, ρ = density of the QUBO

Appendix D TTT supplementary plots

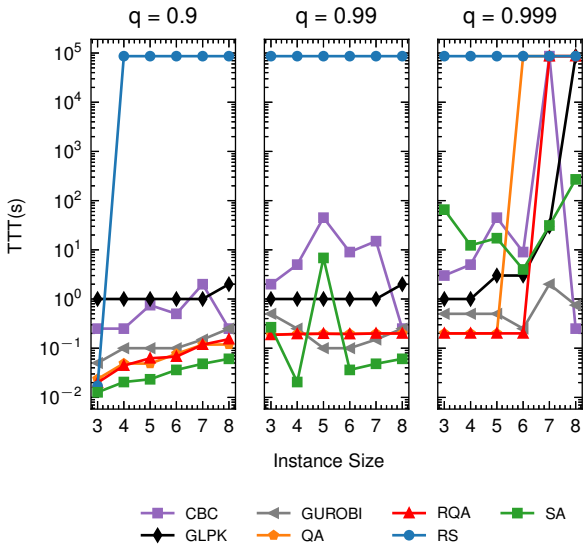


Fig. D2: Time-to-Target, reported in seconds for Cumulative instances with OS = 0.1.

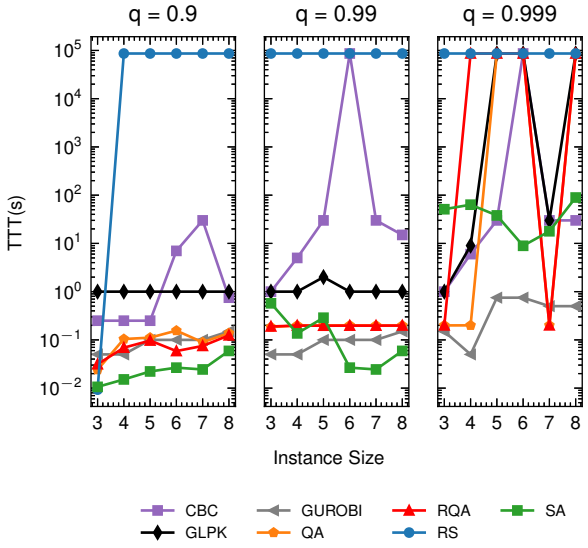


Fig. D3: Time-to-Target, reported in seconds for Medium OS instances with OS = 0.5.

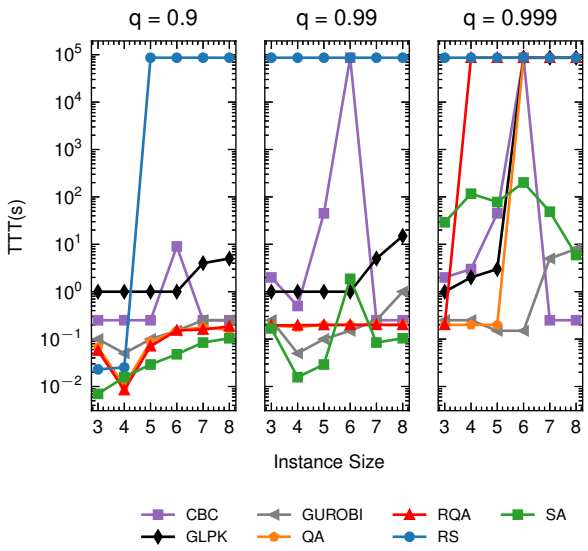


Fig. D4: Time-to-Target, reported in seconds for Disjunctive instances with OS = 0.9.

Appendix E Annealing time supplementary plots

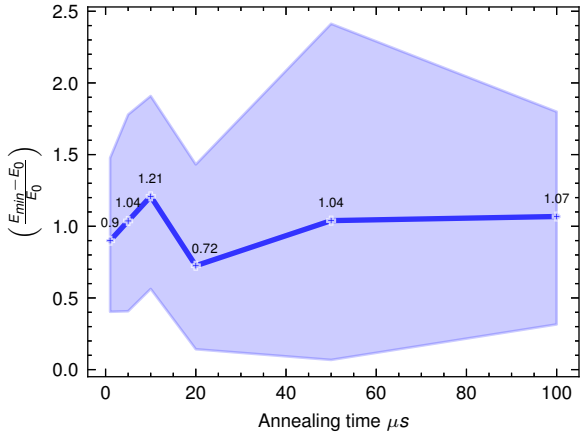


Fig. E5: Overall effect of annealing time on QA performance, measured as the mean relative deviation from the ground state $\left(\frac{E_{\min}-E_0}{E_0}\right)$ for Cumulative instances and all sizes (i.e., 3, 4, 5, 6, 7, and 8 non-dummy activities).

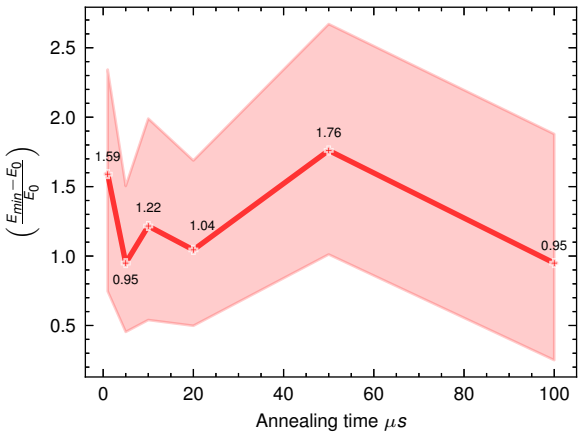


Fig. E6: Overall effect of annealing time on QA performance, measured as the mean relative deviation from the ground state $\left(\frac{E_{\min}-E_0}{E_0}\right)$ for Medium OS instances with various sizes (i.e., 3, 4, 5, 6, 7, and 8 non-dummy activities).

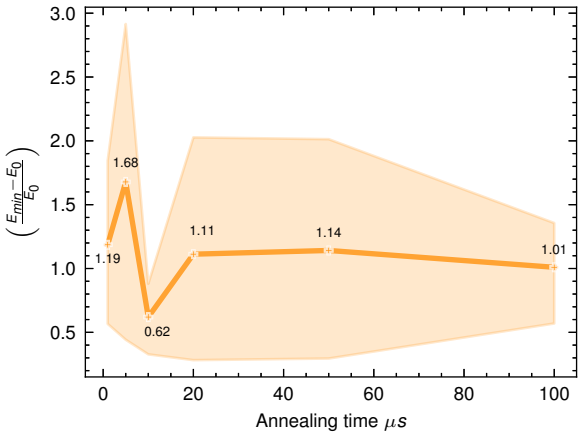


Fig. E7: Overall effect of annealing time on QA performance, measured as the mean relative deviation from the ground state $\left(\frac{E_{\min}-E_0}{E_0}\right)$ for Disjunctive instances and all sizes (i.e., 3, 4, 5, 6, 7, and 8 non-dummy activities).

Appendix F Annealing pauses supplementary plots

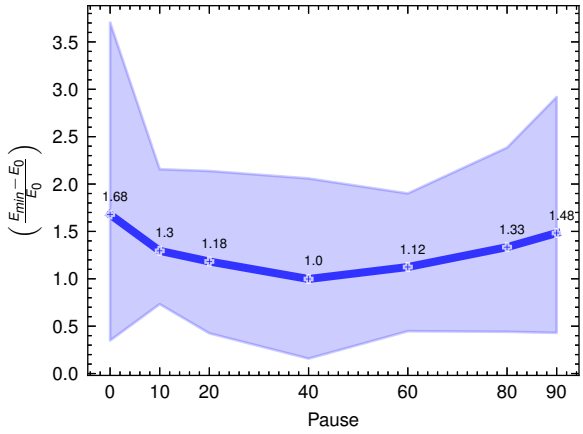


Fig. F8: General effect of pauses included in the annealing schedule vs the relative deviation from the ground state energy $\left(\frac{E_{\min}-E_0}{E_0}\right)$ for Cumulative instances. The y-axis shows the percentage of pause in the annealing schedule, as shown in Fig. 7.

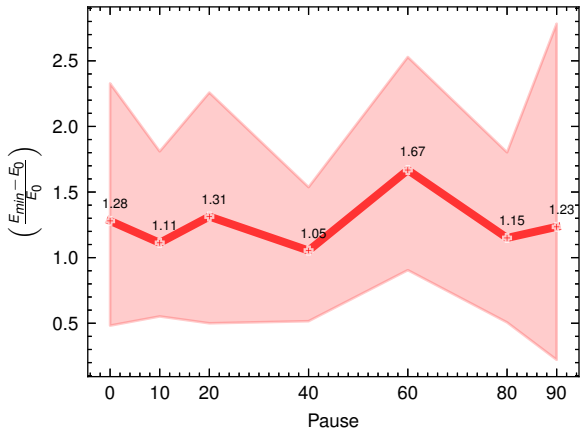


Fig. F9: General effect of pauses included in the annealing schedule vs the relative deviation from the ground state energy $\left(\frac{E_{\min}-E_0}{E_0}\right)$ for Medium OS instances. The y-axis shows the percentage of pause in the annealing schedule, as shown in Fig.7.

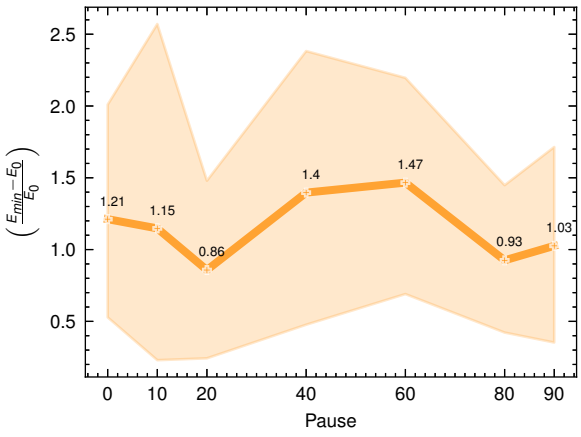


Fig. F10: General effect of pauses included in the annealing schedule vs the relative deviation from the ground state energy $\left(\frac{E_{\min}-E_0}{E_0}\right)$ for Disjunctive instances. The y-axis shows the percentage of pause in the annealing schedule, as shown in Fig.7.

Appendix G QUBO of the 1_PRCPSP

$$
\begin{aligned}
f(x)^{\text{1\_PRCPSP}}_{\text{QUBO}} = & \sum_t t x_{(n+1)t} \\
& + \lambda_1^A \sum_{i=0}^{n+1} \left(1 - \sum_t x_{iAt}\right)^2 \\
& + \lambda_1^B \sum_{i=0}^{n+1} \left(1 - \sum_t x_{iBt}\right)^2 \\
& + \lambda_2^{BB} \sum_{(i,j)\in\mathcal{E}} \sum_t \sum_{t'\setminus t+p_{iB}>t'} x_{iBt} x_{jBt'} \\
& + \lambda_2^{AB} \sum_{i\in\mathcal{A}} \sum_t \sum_{t'\setminus t+p_{iA}>t'} x_{iAt} x_{iBt'} \\
& + \lambda_3^C \sum_t \sum_k \left(\sum_{i=1}^{n} b_{ik} \sum_{\tau=t-p_{iA}+1}^{t} x_{iA\tau} + x_{iB\tau} - B_k + s_{tk}\right)^2 .
\end{aligned}
\tag{G1}
$$

We adapt the notation of the multipliers according to the activity and the type of constraints (e.g. "λ_1^B"). Each original variable x_{it} is now divided into two parts x_{iAt} and x_{iBt} that must comply with:

$$
\sum_{t\in\mathcal{H}} t x_{iAt} \leq \sum_{t\in\mathcal{H}} t x_{iBt} + p_A \qquad \forall i \in \mathcal{A}.
$$

References

[1] Preskill, J.: Quantum computing in the nisq era and beyond. Quantum **2**, 79 (2018)

[2] Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 124–134 (1994). Ieee

[3] Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, pp. 212–219 (1996)

[4] MIT Technology Review, M.I.o.T.: IBM wants to build a 100,000-qubit quantum computer. https://www.technologyreview.com/2023/05/25/1073606/ibm-wants-to-build-a-100000-qubit-quantum-computer/#:~:text=Late%20last%20year%2C%20IBM%20took,blocks%20of%20quantum%20information%20processing. 11-2023 (2023)

[5] Ushijima-Mwesigwa, H., Negre, C.F., Mniszewski, S.M.: Graph partitioning using quantum annealing on the d-wave system. In: Proceedings of the Second International Workshop on Post Moores Era Supercomputing, pp. 22–29 (2017)

[6] Bauckhage, C., Piatkowski, N., Sifa, R., Hecker, D., Wrobel, S.: A qubo formulation of the k-medoids problem. In: LWDA, pp. 54–63 (2019)

[7] Neukart, F., Compostella, G., Seidel, C., Von Dollen, D., Yarkoni, S., Parney, B.: Traffic flow optimization using a quantum annealer. Frontiers in ICT **4**, 29 (2017)

[8] Stollenwerk, T., O'Gorman, B., Venturelli, D., Mandra, S., Rodionova, O., Ng, H., Sridhar, B., Rieffel, E.G., Biswas, R.: Quantum annealing applied to de-conflicting optimal trajectories for air traffic management. IEEE transactions on intelligent transportation systems **21**(1), 285–297 (2019)

[9] Venturelli, D., Marchand, D.J., Rojo, G.: Quantum annealing implementation of job-shop scheduling. arXiv preprint arXiv:1506.08479 (2015)

[10] Carugno, C., Ferrari Dacrema, M., Cremonesi, P.: Evaluating the job shop scheduling problem on a d-wave quantum annealer. Scientific Reports **12**(1), 6539 (2022)

[11] Venturelli, D., Mandrà, S., Knysh, S., O'Gorman, B., Biswas, R., Smelyanskiy, V.: Quantum optimization of fully connected spin glasses. Physical Review X **5**(3), 031040 (2015)

[12] Bian, Z., Chudak, F., Macready, W., Roy, A., Sebastiani, R., Varotti, S.: Solving sat (and maxsat) with a quantum annealer: Foundations, encodings, and

preliminary results. Information and computation **275**, 104609 (2020)

[13] Rosenberg, G., Haghnegahdar, P., Goddard, P., Carr, P., Wu, K., De Prado, M.L.: Solving the optimal trading trajectory problem using a quantum annealer. In: Proceedings of the 8th Workshop on High Performance Computational Finance, pp. 1–7 (2015)

[14] King, A.D., Carrasquilla, J., Raymond, J., Ozfidan, I., Andriyash, E., Berkley, A., Reis, M., Lanting, T., Harris, R., Altomare, F., *et al.*: Observation of topological phenomena in a programmable lattice of 1,800 qubits. Nature **560**(7719), 456–460 (2018)

[15] King, J., Yarkoni, S., Nevisi, M.M., Hilton, J.P., McGeoch, C.C.: Benchmarking a quantum annealing processor with the time-to-target metric. arXiv preprint arXiv:1508.05087 (2015)

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

[17] Albash, T., Lidar, D.A.: Adiabatic quantum computation. Reviews of Modern Physics **90**(1), 015002 (2018)

[18] Kirkpatrick, S., Gelatt Jr, C.D., Vecchi, M.P.: Optimization by simulated annealing. science **220**(4598), 671–680 (1983)

[19] Harris, R., Johansson, J., Berkley, A., Johnson, M., Lanting, T., Han, S., Bunyk, P., Ladizinsky, E., Oh, T., Perminov, I., *et al.*: Experimental demonstration of a robust and scalable flux qubit. Physical Review B **81**(13), 134510 (2010)

[20] Santoro, G.E., Martonák, R., Tosatti, E., Car, R.: Theory of quantum annealing of an ising spin glass. Science **295**(5564), 2427–2430 (2002)

[21] Johnson, M.W., Amin, M.H., Gildert, S., Lanting, T., Hamze, F., Dickson, N., Harris, R., Berkley, A.J., Johansson, J., Bunyk, P., *et al.*: Quantum annealing with manufactured spins. Nature **473**(7346), 194–198 (2011)

[22] Vert, D., Sirdey, R., Louise, S.: Benchmarking quantum annealing against "hard" instances of the bipartite matching problem. SN Computer Science **2**, 1–12 (2021)

[23] Born, M., Fock, V.: Beweis des adiabatensatzes. Zeitschrift für Physik **51**, 165–180 (1928) https://doi.org/10.1007/BF01343193

[24] Kato, T.: On the adiabatic theorem of quantum mechanics. Journal of the Physical Society of Japan **5**(6), 435–439 (1950)

[25] Schrödinger, E.: An undulatory theory of the mechanics of atoms and molecules. Physical review **28**(6), 1049 (1926)

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

[27] Josephson, B.D.: Possible new effects in superconductive tunnelling. Physics letters **1**(7), 251–253 (1962)

[28] Kwon, S., Tomonaga, A., Lakshmi Bhai, G., Devitt, S.J., Tsai, J.-S.: Gate-based superconducting quantum computing. Journal of Applied Physics **129**(4), 041102 (2021)

[29] Krantz, P., Kjaergaard, M., Yan, F., Orlando, T.P., Gustavsson, S., Oliver, W.D.: A quantum engineer's guide to superconducting qubits. Applied physics reviews **6**(2), 021318 (2019)

[30] Bernal, D.E., Booth, K.E., Dridi, R., Alghassi, H., Tayur, S., Venturelli, D.: Integer programming techniques for minor-embedding in quantum annealers. In: Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 17th International Conference, CPAIOR 2020, Vienna, Austria, September 21–24, 2020, Proceedings 17, pp. 112–129 (2020). Springer

[31] Okada, S., Ohzeki, M., Terabe, M., Taguchi, S.: Improving solutions by embedding larger subproblems in a d-wave quantum annealer. Scientific reports **9**(1), 2098 (2019)

[32] Abbott, A.A., Calude, C.S., Dinneen, M.J., Hua, R.: A hybrid quantum-classical paradigm to mitigate embedding costs in quantum annealing. International Journal of Quantum Information **17**(05), 1950042 (2019)

[33] Serra, T., Huang, T., Raghunathan, A.U., Bergman, D.: Template-based minor embedding for adiabatic quantum optimization. INFORMS Journal on Computing **34**(1), 427–439 (2022)

[34] Choi, V.: Minor-embedding in adiabatic quantum computation: I. the parameter setting problem. Quantum Information Processing **7**, 193–209 (2008)

[35] Venturelli, D., Kondratyev, A.: Reverse quantum annealing approach to portfolio optimization problems. Quantum Machine Intelligence **1**(1-2), 17–30 (2019)

[36] Ikeda, K., Nakamura, Y., Humble, T.S.: Application of quantum annealing to nurse scheduling problem. Scientific reports **9**(1), 12837 (2019)

[37] Rocutto, L., Destri, C., Prati, E.: Quantum semantic learning by reverse annealing of an adiabatic quantum computer. Advanced Quantum Technologies **4**(2), 2000133 (2021)

[38] Golden, J., O'Malley, D.: Reverse annealing for nonnegative/binary matrix

factorization. Plos one **16**(1), 0244026 (2021)

[39] Blazewicz, J., Lenstra, J.K., Kan, A.R.: Scheduling subject to resource constraints: classification and complexity. Discrete applied mathematics **5**(1), 11–24 (1983)

[40] Sahni, S., Cho, Y.: Complexity of scheduling shops with no wait in process. Mathematics of Operations Research **4**(4), 448–457 (1979)

[41] Gromicho, J.A., Van Hoorn, J.J., Saldanha-da-Gama, F., Timmer, G.T.: Solving the job-shop scheduling problem optimally by dynamic programming. Computers & Operations Research **39**(12), 2968–2977 (2012)

[42] Garey, M.R.: Computers and intractability: A guide to the theory of np-completeness, freeman. Fundamental (1997)

[43] Boland, N., Christiansen, J., Dandurand, B., Eberhard, A., Oliveira, F.: A parallelizable augmented lagrangian method applied to large-scale non-convex-constrained optimization problems. Mathematical Programming **175**, 503–536 (2019)

[44] Castro, J., Nasini, S., Saldanha-da-Gama, F.: A cutting-plane approach for large-scale capacitated multi-period facility location using a specialized interior-point method. Mathematical programming **163**(1-2), 411–444 (2017)

[45] Castro, J., Nasini, S.: A specialized interior-point algorithm for huge minimum convex cost flows in bipartite networks. European Journal of Operational Research **290**(3), 857–869 (2021)

[46] Damay, J., Quilliot, A., Sanlaville, E.: Linear programming based algorithms for preemptive and non-preemptive rcpsp. European Journal of Operational Research **182**(3), 1012–1022 (2007)

[47] Brucker, P., Knust, S.: A linear programming and constraint propagation-based lower bound for the rcpsp. European Journal of Operational Research **127**(2), 355–362 (2000)

[48] Pritsker, A.A.B., Waiters, L.J., Wolfe, P.M.: Multiproject scheduling with limited resources: A zero-one programming approach. Management science **16**(1), 93–108 (1969)

[49] Christofides, N., Alvarez-Valdés, R., Tamarit, J.M.: Project scheduling with resource constraints: A branch and bound approach. European journal of operational research **29**(3), 262–273 (1987)

[50] De Souza, C., Wolsey, L.: Scheduling projects with labour constraints. Relatório Técnico IC-P7-22. Instituto de Computação, Universidade Estadual de Campinas

(1997)

[51] Mingozzi, A., Maniezzo, V., Ricciardelli, S., Bianco, L.: An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. Management science **44**(5), 714–729 (1998)

[52] Klein, R.: Scheduling of Resource-constrained Projects vol. 10. Springer, New York, USA (2000)

[53] Kaplan, L.A.: Resource-constrained Project Scheduling with Preemption of Jobs. University of Michigan, Michigan, USA (1988)

[54] Bianco, L., Caramia, M.: A new formulation for the project scheduling problem under limited resources. Flexible Services and Manufacturing Journal **25**, 6–24 (2013)

[55] Olaguibel, R.A.-V., Goerlich, J.T.: The project scheduling polyhedron: Dimension, facets and lifting theorems. European Journal of Operational Research **67**(2), 204–220 (1993)

[56] Artigues, C., Michelon, P., Reusser, S.: Insertion techniques for static and dynamic resource-constrained project scheduling. European Journal of Operational Research **149**(2), 249–267 (2003)

[57] Koné, O., Artigues, C., Lopez, P., Mongeau, M.: Event-based milp models for resource-constrained project scheduling problems. Computers & Operations Research **38**(1), 3–13 (2011)

[58] Artigues, C., Brucker, P., Knust, S., Koné, O., Lopez, P., Mongeau, M.: A note on "event-based milp models for resource-constrained project scheduling problems". Computers & Operations Research **40**(4), 1060–1063 (2013)

[59] Koné, O., Artigues, C., Lopez, P., Mongeau, M.: Comparison of mixed integer linear programming models for the resource-constrained project scheduling problem with consumption and production of resources. Flexible Services and Manufacturing Journal **25**, 25–47 (2013)

[60] Artigues, C., Demassey, S., Neron, E.: Resource-constrained Project Scheduling: Models, Algorithms, Extensions and Applications. Wiley, New Jersey, USA (2013)

[61] Kolisch, R., Sprecher, A.: Psplib-a project scheduling problem library: Or software-orsep operations research software exchange program. European journal of operational research **96**(1), 205–216 (1997)

[62] Kolisch, R., Sprecher, A.: PSPLIB. https://www.om-db.wi.tum.de/psplib/data.html. 11-2023 (1996)

[63] Coelho, J., Vanhoucke, M.: Going to the core of hard resource-constrained project scheduling instances. Computers & Operations Research **121**, 104976 (2020)

[64] Schoot, W., Leermakers, D., Wezeman, R., Neumann, N., Phillipson, F.: Evaluating the q-score of quantum annealers. In: 2022 IEEE International Conference on Quantum Software (QSW), pp. 9–16 (2022). IEEE

[65] Demeulemeester, E., Vanhoucke, M., Herroelen, W.: Rangen. https://www.projectmanagement.ugent.be/research/data/RanGen. 08-2023 (2003)

[66] Baptiste, P., Pape, C.L.: Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems. Constraints **5**, 119–139 (2000)

[67] Lidar, D.A., Wecker, D., Martinis, J.M., Job, J., Troyer, M., Isakov, S.V., Boixo, S., Wang, Z., et al.: Defining and detecting quantum speedup. Science (2014)

[68] Martiel, S., Ayral, T., Allouche, C.: tbenchmarking quantum coprocessors in an application-centric, hardware-agnostic, and scalable way. IEEE Transactions on Quantum Engineering **2**, 1–11 (2021)

[69] Farhi, E., Goldstone, J., Gutmann, S.: A quantum approximate optimization algorithm. arXiv preprint arXiv:1411.4028 (2014)

[70] Lucas, A.: Ising formulations of many np problems. Frontiers in physics **2**, 5 (2014)

[71] Verma, A., Lewis, M.: Penalty and partitioning techniques to improve performance of qubo solvers. Discrete Optimization **44**, 100594 (2022)

[72] Demeulemeester, E.L., Herroelen, W.S.: Project Scheduling: a Research Handbook vol. 49. Springer, New York, USA (2006)

[73] Marshall, J., Venturelli, D., Hen, I., Rieffel, E.G.: Power of pausing: Advancing understanding of thermalization in experimental quantum annealers. Physical Review Applied **11**(4), 044083 (2019)

[74] Ballestín, F., Valls, V., Quintanilla, S.: Pre-emption in resource-constrained project scheduling. European Journal of Operational Research **189**(3), 1136–1152 (2008)

[75] Hartmann, S.: Project scheduling with resource capacities and requests varying with time: a case study. Springer (2013)

[76] Kolisch, R., Hartmann, S.: Experimental evaluation of heuristics for the resource-constrained project scheduling problem: An update. European Journal of Operational Research (2005)

[77] Sadeghi, H.: Quantum computing with neutral atoms with applications in energy. In: EAGE Seventh High Performance Computing Workshop, vol. 2023, pp. 1–3 (2023). European Association of Geoscientists & Engineers

Chapter 5

cheche-pm: A Python Library for Project Scheduling

The content of this chapter is presented in the form of a paper titled "**cheche_pm: a dedicated python library for project scheduling and project management**".

5.1 Chapter 5 introduction

In this chapter, we explore the development and evolution of Cheche-PM, an open-source Python library engineered for project scheduling. That was pivotal for the development of the work of chapter 4. This library is specifically designed to navigate two main challenges that often appear in project management: the scheduling of projects under resource constraints and the scheduling of projects amidst uncertainty.

CHECHE_PM: A DEDICATED PYTHON LIBRARY FOR PROJECT SCHEDULING AND PROJECT MANAGEMENT *

Luis Fernando Pérez Armas
Operations management, IESEG School of Management Univ. Lille, CNRS, UMR 9221 - LEM
Lille - France
l.perezarmas@ieseg.fr

ABSTRACT

Project scheduling is critical for the timely and efficient completion of projects, a task complicated by resource constraints and the unpredictable nature of project activities. The limited availability of comprehensive and affordable commercial software, which often overlooks recent advances in scheduling techniques, further exacerbates these challenges. To bridge this gap, we present `cheche_pm`, an open-source Python library that incorporates a collection of advanced and current scheduling methods. Designed for practitioners, students, and academics alike, `cheche_pm` offers user-friendly APIs for those without specialist knowledge, modular components for developers, extensive documentation for learners, and a wide array of methods to meet diverse needs. To the best of our knowledge, `cheche_pm` represents the first Python library dedicated exclusively to project scheduling, constituting a significant advancement in the discipline and opening the door for a more broader audience interested in this domain.

***K*eywords** Project Management · Project Planning · Project Scheduling · Python

1 Introduction

The practice of project management has been an essential aspect of human nature for millennia, dating back to the construction of ancient wonders like the pyramids in Egypt and Maya temples in Central America. These historical feats showcase early instances of project management, a field that has since evolved into a critical discipline for organizing and executing complex tasks. The concept of project management, as delineated by Badiru [1], encompasses the efficient and timely allocation of resources to meet project objectives. This field's importance is underscored by its rapid growth and the Project Management Institute's (PMI) forecast of nearly 90 million project management-related jobs by 2027 [2]. Highlighting this significance, in 2017, the renowned magazine The Economist estimated that construction projects alone contributed approximately 11 trillion USD to the global GDP.

A project, as defined by the Project Management Institute, is *a temporary endeavor with the goal of creating a unique product, service, or result*. This temporary and unique nature distinguishes projects from traditional, ongoing business operations, as noted by Dinsmore and Cooke-Davies [3]. Projects are complex, one-off undertakings involving a series of interrelated activities, each consuming time, resources, and incurring costs. These activities are often interdependent due to technical or organizational requirements and resource constraints, necessitating careful scheduling and management.

Effective project management, therefore, involves meticulous project planning, a phase for which the importance is highlighted by the works of Shenhar et al. [4, 5, 6], Blomquist et al. [7], Dvir et al. [8], and finally Turner et al. [9] İn this phase, projects are broken down into activities, with detailed planning of each activity's duration, required resources, costs, and precedence relations. Project scheduling emerges as a pivotal aspect of project planning, aiming to allocate resources and schedule activities within the constraints of time and resources. The complexity of project scheduling

* *Citation*: **Luis Fernando Pérez Armas. cheche_pm: a dedicated python library for project scheduling and project management.**

varies, ranging from time-constrained problems, where resources are assumed unlimited, to resource-constrained problems, where resource availability is limited.

Historically, formal methods in project management gained significant traction during the 1950s and 1960s with large-scale projects like the Polaris Missile program and the Apollo moon landing [10]. These projects primarily dealt with time-constrained scheduling, giving rise to methodologies like the critical path method. However, the assumption of unlimited resources, common in these methodologies, is often unrealistic in practical scenarios, leading to the acknowledgment of resource-constrained project scheduling as a more challenging and NP-hard problem as demonstrated by Błażejewicz et al. [11].

Project scheduling objectives extend beyond minimizing project duration, like optimizing the net present value (NPV) of a project[12], or addressing the stochastic nature of project durations and costs. Despite the importance and complexity of these scheduling problems, most commercial project management software fails to provide comprehensive and affordable solutions, especially for individual practitioners and students.

To bridge this gap, this paper introduces `cheche_pm` an open-source Python library dedicated to project scheduling. `Cheche_pm` features a wide array of scheduling algorithms, user-friendly APIs, and extensive documentation, making it accessible to a diverse user base. Its modular design allows for specialized functionalities, including constructive heuristics, utility functions, quantitative risk management, and metaheuristic-based optimizers. Unique in its complete Python implementation, `cheche_pm` integrates seamlessly with the vast Python ecosystem, enabling users without expertise in other languages like C++, Java, or R to leverage its capabilities efficiently. This library not only facilitates the application of advanced scheduling techniques but also encourages the development of novel methods, significantly contributing to the field of project scheduling.

2 Design

`Cheche_pm` is conceived as an open-source, modular Python platform, streamlining extensibility and integration. It addresses the complexity newcomers face in project scheduling due to the diversity of methods and their varying assumptions and goals. The library succinctly encapsulates a range of algorithms and tools, emphasizing practical applicability and context-specific utility. This design not only broadens accessibility to advanced scheduling techniques but also clarifies their strategic implementation in project management.

2.1 Project data input

`Cheche_pm` is designed to offer versatile and user-friendly methods for inputting and creating project instances, catering to both small-scale and large-scale project requirements. Initially, users have the option to manually create an empty project instance and populate it incrementally by adding activities. For each activity, essential details such as duration, precedence constraints, and resource consumption for each resource must be specified. Additionally, the user needs to define the maximum resource capacity for each resource. Optionally, activity-specific costs can also be provided.

While this manual method is convenient for smaller projects, it can be time-consuming for larger ones. To streamline the process for more extensive projects, `cheche_pm` facilitates project instance creation from both CSV and Excel (.xlsx) files. These files should encompass all the columns necessary for manual creation, effectively automating the data input process for each activity.

For further convenience and to accommodate a wider range of data sources, `cheche_pm` includes the capability to directly import project data from .rcp files. This feature is particularly useful for academic users working with established datasets, such as those from the Kolisch et al . PSPLIB dataset [13] or the Vanhoucke et al CV dataset [14].

Moreover, `cheche_pm` is equipped with the functionality to export project data in both CSV and Excel formats. Looking ahead, one of the key features on the development roadmap is the inclusion of output capability in the .rcp format.

2.2 Resource Unconstrained Project Scheduling Problem (RUPSP)

`Cheche_pm` features a function, `.CPM()` which implements the Critical Path Method (CPM) [15], suitable to solve resource unconstrained project scheduling problems. This function efficiently calculates and outputs key metrics for each project activity, including Early Start, Early Finish, Late Start, Late Finish, and float times. An innovative aspect of this function is the “verbose" input argument, specifically designed for educational purposes. When set to **TRUE** this argument activates detailed logging of each algorithmic decision, providing an invaluable tool for students to learn and understand the inner workings of the CPM algorithm. Additionally, the library includes a function, `.get_critical_path()`, which identifies and returns the critical activities within a project. This dual functionality of

cheche_pm makes it a powerful resource for both practical project management applications and academic learning environments.

2.3 Visualization functions

Project visualizations play a pivotal role in the planning, execution, monitoring, and controlling phases of project management. Among the most renowned visualization tools are network diagrams, Gantt charts, and resource level plots. cheche_pm caters to this need by offering functionalities to visualize project networks, highlighting precedence relationships. Users can choose between networkx plots or pydot plots, and future updates will include interactive network plots using pygraphviz in HTML format. The Gantt chart, a widely used visualization tool in project management, is another key feature of cheche_pm. The library enables the creation of Gantt charts for any project schedule, with output options including both a traditional format using matplotlib and an interactive format using plotly. These charts can be exported in various image formats such as PNG, JPG, JPEG, SVG, PDF, and HTML for plotly plots. cheche_pm enhances flexibility by allowing Gantt charts to be generated with either relative dates or absolute datetimes, accommodating various datetime formats through the integration of pandas' datetime reader. Additionally, cheche_pm provides the capability to produce resource level plots. These plots are instrumental in identifying potential resource constraint violations in project schedules, offering a crucial tool for efficient project management.

2.4 Heuristics for solving the RCPSP

Cheche_pm is equipped with an extensive range of heuristics specifically designed to tackle the Resource Constrained Project Scheduling Problem (RCPSP). At its foundation, the library enables the creation of constructive priority lists using various criteria as the ruling priority rule [16, 17, 18]. These rules can be effectively transformed into schedules through two distinct scheduling algorithms: the sequential schedule generation algorithm (SGS) [19] and the parallel schedule generation algorithm (PGS). Echoing the functionality seen in the .CPM function, both scheduling algorithms, as well as the priority rule function, include a "verbose" input option. When set to **TRUE**, this feature provides detailed logs of the algorithmic decisions, facilitating a learning environment for students grappling with these complex problems. Reflecting its commitment to educational utility, cheche_pm places a strong emphasis on fostering understanding and skill development in project scheduling.

Additionally, cheche_pm offers a unique function that allows users to execute all available heuristics, aiming to generate the best feasible schedule. This function not only produces the optimal schedule but also identifies the specific priority rule and scheduling method responsible for its creation, offering valuable insights into the decision-making process.

Looking forward, the short-term development roadmap for cheche_pm includes the integration of reverse scheduling heuristics, forward and backward scheduling [20], and the justification algorithm [21]. The mid-term roadmap is set to expand its capabilities further, with the inclusion of constructive heuristics tailored for the RCPSP with discounted cash flows (RCPSPDC) that maximizes the net-present value of a project instead of its make-span [22], the preemptive-RCPSP [23], and the Stochastic Resource Constrained Project Scheduling Problem (SRCPSP) [18, 24].

2.5 Meta-heuristic optimizers for project scheduling

Cheche_pm leverages the efficacy of constructive heuristics for the RCPSP, which have demonstrated remarkable performance, often approaching within 6% of optimal solutions [16] in challenging datasets like PSPLIB and CV. For users seeking even closer approximations to optimal solutions, cheche_pm provides access to various metaheuristic optimizers. The library currently includes a genetic algorithm tailored for the RCPSP. In the near future, enhancements to the genetic optimizer will enable it to tackle the RCPSPDC and other variations of the scheduling problem.

Additionally, cheche_pm features an implementation of simulated annealing with a re-heating option [25], further broadening its optimization capabilities. This metaheuristic is currently available for the RCPSP, with plans to extend its applicability to other RCPSP variants. The mid-term development roadmap of cheche_pm is particularly ambitious, aiming to incorporate a suite of advanced metaheuristics. This expansion will include particle-swarm optimization, ant colony optimization, tabu-search, and the neuro-genetic algorithm, significantly enriching the library's problem-solving arsenal and positioning it as a versatile tool for both academic research and practical application in complex project scheduling scenarios.

2.6 Quantitative project risk analysis

Cheche_pm encompasses a range of methods for conducting quantitative risk analysis, crucial for effective project management. A notable feature is the integration of the Project Evaluation and Review Technique (PERT), pioneered

by the US Navy in the 1960s [26], which offers project duration and cost estimations under uncertainty using a beta distribution. Despite its utility, PERT has limitations, notably its disregard for resource constraints and the assumption of a constant critical path, which is rarely the case in practice.

To address these limitations, `cheche_pm` introduces the capability for Monte Carlo simulations, enhancing the realism and accuracy of risk assessment. This method models each activity with a triangular distribution, defined by optimistic, pessimistic, and most likely duration values. The output from these simulations provides valuable metrics such as the Criticality Index (CI), Significance Index (SI), Cruciality Index (CRI), and others [27]. It also estimates the probability of activities starting as scheduled based on most likely values and yields 95% and 99% confidence schedules.

Further, the library enables buffer analysis, allowing users to input buffer locations and durations. The Monte Carlo simulation then calculates new on-time start probabilities, helping users understand the impact of protecting various activities. Looking ahead, `ccheche_pm` development roadmap includes advanced quantitative risk analysis features. These will incorporate the CDCA and CDCR indices from the work of CREEMERS and Demeulemeester [28], offering finer-grained risk information and robust prioritization. Additionally, plans are underway to extend Monte Carlo simulations to the SRCPSP and the Stochastic RCPSP with Discounted Cash Flows (SRCPSPDC), further expanding the library's risk analysis capabilities.

2.7 Utility functions

`Cheche_pm` is not only a robust tool for project scheduling but also offers a suite of utility functions that are particularly beneficial for academic research. One of its key features is the generation of Python-friendly data structures, which can be instrumental in formulating Mixed Integer Linear Programming (MILP) models for the RCPSP, SRCPSP, and RCPSPDC. These data structures facilitate the integration of complex project scheduling models into computational frameworks.

Moreover, `cheche_pm` is equipped to compute specific input information vital for certain MILP formulations. For example, it enables the calculation of "Forbidden Sets," a necessary component for the Tamarit and Valdes (1993) [29] formulation, and "Feasible Sets," required for the Mingozzi et al. (1998) [30] formulation. These functionalities underscore `cheche_pm's` value in addressing advanced scheduling problems and contributing to research in this field.

Looking to the future, the development roadmap for `cheche_pm` includes features that will further enhance its utility and interoperability. A notable addition will be the capability to import and export data in Microsoft Project-compatible .xml files. This feature will significantly increase the ease of integrating `cheche_pm` with mainstream project management software. Additionally, the roadmap includes plans to introduce interactive Work Breakdown Structure (WBS) tree diagrams and automatic PDF project report generation, providing users with more dynamic and comprehensive tools for project visualization and documentation.

3 Example

In this section, let us demonstrate how to use `cheche_pm` for project scheduling. The library can easily installed from `Pypi` (`https://pypi.org/project/cheche-pm/`) by using the following command below:

```
!pip install cheche_pm
```

Once the library is installed the user is ready to import it and start using it. The easiest way is to import the blank Project instance from `cheche_pm` and start populating it.

```
from cheche_pm import Project
```

For the rest of this example, the project example from Table 1 will be used.

idx	description	duration	precedence	resources	cost
Start	Start of Project	0	None	[0, 0, 0]	0
A	F.House	5	[Start]	[1, 0, 0]	0
B	F.Pool	2	[Start]	[1, 0, 0]	0
C	Walls	5	[A]	[0, 1, 0]	0
D	Pool	6	[B]	[0, 0, 1]	0
E	Roof	5	[C]	[0, 1, 0]	0
F	Windows	2	[C]	[0, 1, 0]	0
G	Electricity	3	[C]	[0, 0, 1]	0
H	S.Panels	2	[E]	[0, 0, 1]	0
I	Plumbing	4	[F]	[0, 0, 1]	0
J	Finishings	3	[H, I]	[0, 0, 1]	0
End	End of Project	0	[D, G, J]	[0, 0, 0]	0

Table 1: Project Example

The Project instance from `cheche_pm` can be manually populated as it is demonstrated in the code below:

```
p = Project()

p.add_activity(activity_name='A',activity_duration = 5,
activity_precedence= [None], a_desc= 'F.House', activity_resources=[1,0,0])

p.add_activity(activity_name='B',activity_duration = 2,
activity_precedence= [None], a_desc= 'F.Pool', activity_resources=[1,0,0])

p.add_activity(activity_name='C',activity_duration = 5,
activity_precedence= ['A'], a_desc= 'Walls', activity_resources=[0,1,0])

p.add_activity(activity_name='D',activity_duration = 6,
activity_precedence= ['B'], a_desc= 'Pool', activity_resources=[0,0,1])

p.add_activity(activity_name='E',activity_duration = 5,
activity_precedence= ['C'], a_desc= 'Roof', activity_resources=[0,1,0])

p.add_activity(activity_name='F',activity_duration = 2,
activity_precedence= ['C'], a_desc= 'Windows', activity_resources=[0,1,0])

p.add_activity(activity_name='G',activity_duration = 3,
activity_precedence= ['C'], a_desc= 'Electricity', activity_resources=[0,0,1])

p.add_activity(activity_name='H',activity_duration = 2,
activity_precedence= ['E'], a_desc= 'S.Panels', activity_resources=[0,1,0])

p.add_activity(activity_name='I',activity_duration = 4,
activity_precedence= ['F'], a_desc= 'Plumbing', activity_resources=[0,1,0])

p.add_activity(activity_name='J',activity_duration = 3,
activity_precedence= ['H','I'], a_desc= 'Finishings', activity_resources=[0,1,0])
```

In projects encompassing numerous activities, manually inputting each activity into the project can be impractical. To address this, `cheche_pm` provides a feature for importing project data directly from a .csv or an Excel (.xlsx) file. For seamless integration, these files must adhere to a specific format, containing columns with headers and information exactly as illustrated in Table 1.

```
    p = Project.from_csv(filename='data_project.csv', rcpsp_format=True,
                         n_resources= 3, max_resources=[1,1,1])
```

Once the Project instance is created and the data is filled, the user can proceed to create and visualize a network diagram, by following the code below:

```
p.plot_network_diagram(plot_type = 'dot')
```

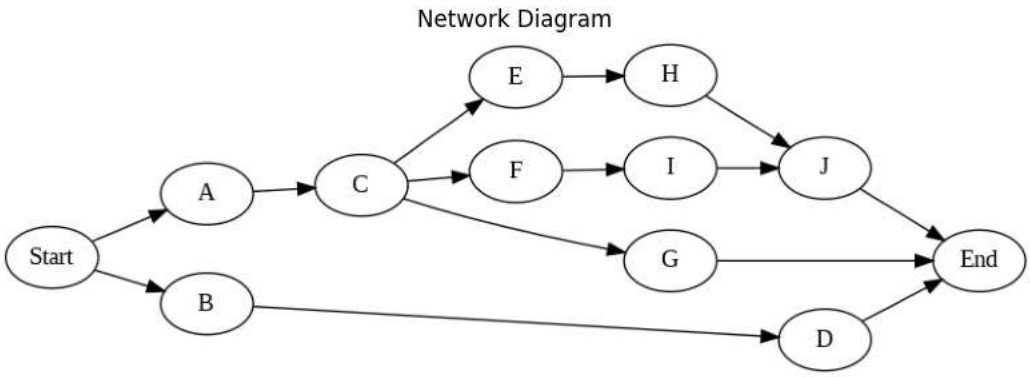


Figure 1: Network diagram of project defined by activities in Table 1 created by `cheche_pm` using “pydot" format.

The user can effortlessly apply the critical path method to derive the results displayed in Table 2. The necessary steps are outlined in the following code snippet:

```
CPM = p.CPM(verbose=True)
CPM
```

	ES	EF	LS	LF	F	D
Start	0	0	0	0	0	0
A	0	5	0	5	0	5
B	0	2	12	14	12	2
C	5	10	5	10	0	5
D	2	8	14	20	12	6
E	10	15	10	15	0	5
F	10	12	11	13	1	2
G	10	13	17	20	7	3
H	15	17	15	17	0	2
I	12	16	13	17	1	4
J	17	20	17	20	0	3
End	20	20	20	20	0	0

Table 2: Outputs from the critical path method of `cheche_pm` for the project described by Table 1

Determining the critical activities is a straightforward process, as demonstrated by executing the following code:

```
p.get_critical_path()
>>>
['Start', 'A', 'C', 'E', 'H', 'J', 'End']
```

The user can then seamlessly generate a Gantt chart to visualize the critical path schedule, as illustrated in the subsequent code (Activities colored in red are critical activities):

```
p.plot_gantt_cpm(early = True)
```

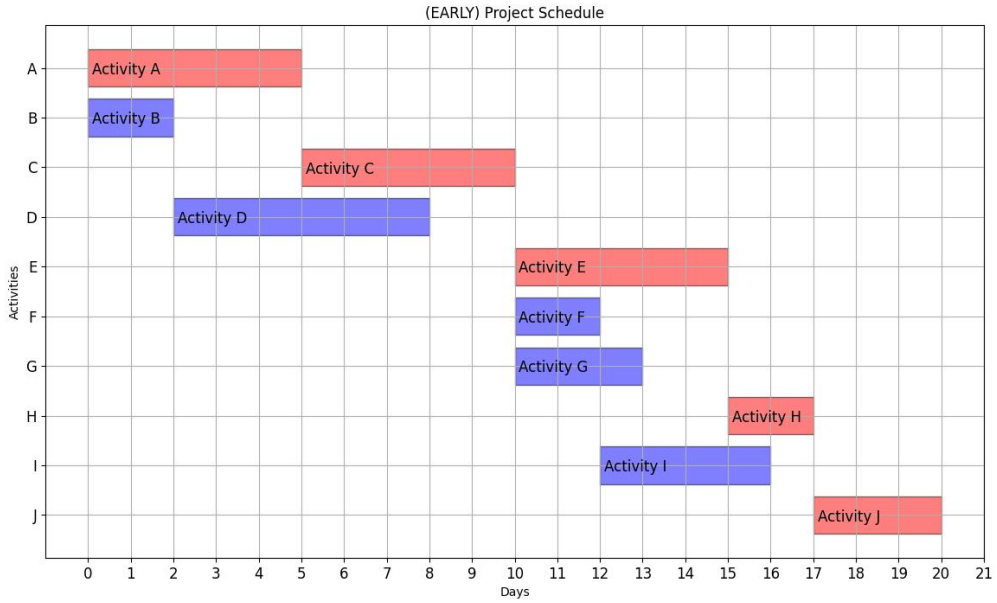


Figure 2: Gantt chart of the critical path schedule presented in Table 2

As previously noted, the project outlined in Table 1 presents a resource-constrained challenge, rendering the critical path method insufficient. In response, `Cheche_pm` offers a feature that enables users to create a priority list according to a selected priority rule. Following this, the tool can generate a schedule based on the derived priority list. This process is exemplified in the code snippet below, which employs the Shortest Processing Time (SPT) priority rule:

```
SPT = p.get_priority_list(priority_rule ='SPT',verbose=False)
SPT
>>>
['Start', 'B', 'A', 'C', 'F', 'G', 'I', 'E', 'H', 'J', 'D', 'End']
```

The user has the option to create either a sequential or a parallel schedule based on the generated priority list, as demonstrated in the following code:

```
SSG = p.SSG(PL = SPT,max_resources=[1,1,1],verbose=True)
SSG
```

```
PSG = p.PSG(PL = SPT,max_resources=[1,1,1],verbose=True)
PSG
```

In the case of the project outlined in Table 1 and employing the “SPT" priority rule, both scheduling techniques yield identical results (in fact this priority rule yields the optimal solution for a final project make-span of 24 days), which are detailed in Table 3.

	Start	B	A	C	F	G	I	E	H	J	D	End
Start Date	0	0	2	7	12	12	15	14	19	21	2	24
End Date	0	2	7	12	14	15	19	19	21	24	8	24

Table 3: SPT schedule obtained from both the sequential and the parallel scheduling methods

Tailored for practitioners, `cheche_pm` enables users to apply the entire suite of priority rules included in the library to both scheduling methods. The function outputs a JSON-formatted dictionary, detailing the optimal schedule alongside the priority rule and scheduling method that produced it. Additionally, it generates a comparison bar chart displaying the project makespan derived from all the heuristic approaches. This functionality can be effortlessly accessed and executed as illustrated in the following code snippet:

```
best_of_all = p.run_all_pl_heuristics(max_resources=[1,1,1])
best_of_all
>>>
        {'PR': 'SPT',
         'schedule_method': 'SSG',
         'solution': {'Start': {'ES': 0, 'EF': 0},
          'B': {'ES': 0, 'EF': 2},
          'A': {'ES': 2, 'EF': 7},
          'C': {'ES': 7, 'EF': 12},
          'F': {'ES': 12, 'EF': 14},
          'G': {'ES': 12, 'EF': 15},
          'I': {'ES': 15, 'EF': 19},
          'E': {'ES': 14, 'EF': 19},
          'H': {'ES': 19, 'EF': 21},
          'J': {'ES': 21, 'EF': 24},
          'D': {'ES': 2, 'EF': 8},
          'End': {'ES': 24, 'EF': 24}},
         'makespan': 24}
```

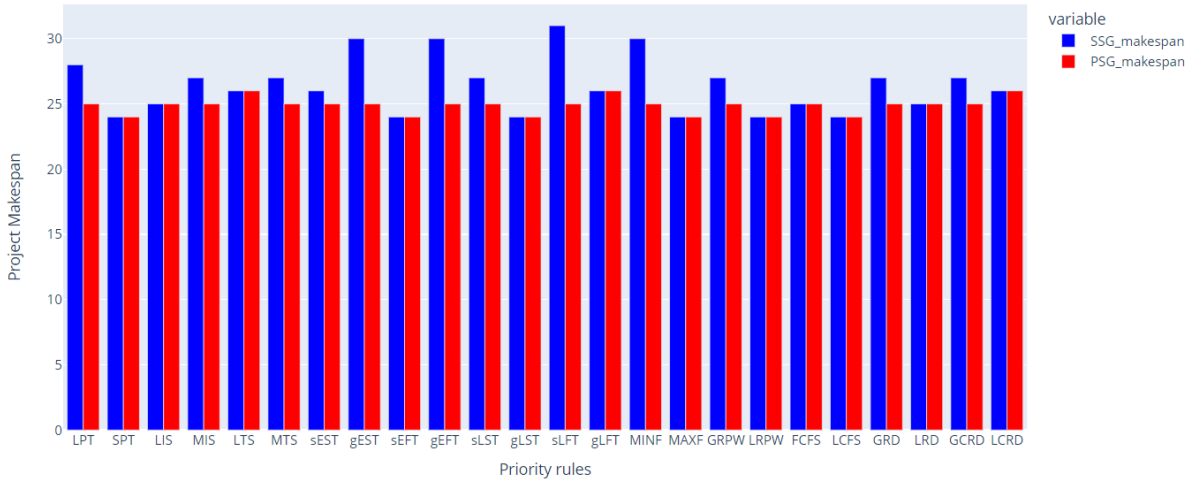


Figure 3: Make-span comparison from different scheduling heuristics applied to the project described by Table 1

The user can conveniently convert the schedule from relative dates to absolute datetimes by simply specifying the project's start date. The library accommodates various datetime formats, defaulting to %Y-%m-%d. Users have the flexibility to include weekends as working days or adjust the schedule to recognize Saturdays and Sundays as

non-working days. The resulting schedule is presented in Table 4. This transformation is straightforwardly achieved with the code provided below:

```
date_time_schedule = p.generate_datetime_schedule(solution=SSG,
                                                  start_date='2023-10-9',
                                                  weekends_work=True,
                                                  verbose=False)
```

	precedence	ES	EF	ES_date	EF_date	ES_weekday	EF_weekday	cal_d	critical
Start	0	0	0	2023-10-09	2023-10-09	Monday	Monday	0	False
B	Start	0	2	2023-10-09	2023-10-11	Monday	Wednesday	2	True
A	Start	2	7	2023-10-11	2023-10-16	Wednesday	Monday	5	True
C	A	7	12	2023-10-16	2023-10-21	Monday	Saturday	5	True
F	C	12	14	2023-10-21	2023-10-23	Saturday	Monday	2	True
G	C	12	15	2023-10-21	2023-10-24	Saturday	Tuesday	3	True
I	F	15	19	2023-10-24	2023-10-28	Tuesday	Saturday	4	True
E	C	14	19	2023-10-23	2023-10-28	Monday	Saturday	5	True
H	E	19	21	2023-10-28	2023-10-30	Saturday	Monday	2	True
J	H,I	21	24	2023-10-30	2023-11-02	Monday	Thursday	3	True
D	B	2	8	2023-10-11	2023-10-17	Wednesday	Tuesday	6	False
End	D,G,J	24	24	2023-11-02	2023-11-02	Thursday	Thursday	0	False

Table 4: Schedule with Absolute Datetimes

This schedule can be seamlessly converted into the Gantt chart depicted in Figure 4. In this chart, activities highlighted in red represent components of the critical chain. It's important to note that in the context of a resource-constrained project, the concept of a 'critical path' becomes obsolete. The Gantt chart can be generated using the following code:

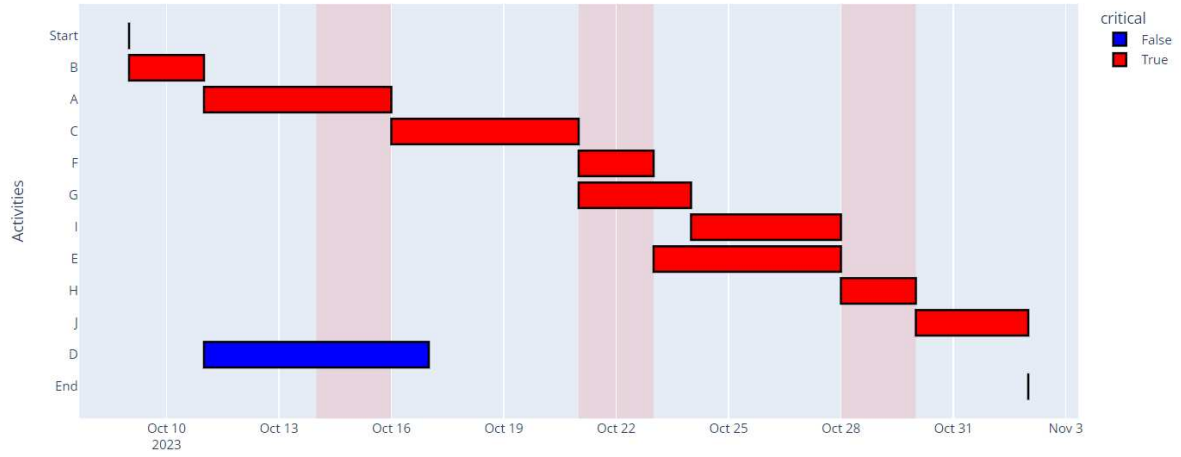


Figure 4: Date time gantt chart from the schedule depicted in Table 4

Utilizing the same deterministic project instance, `cheche_pm` enables the user to account for uncertainties in activity durations by modeling them with a triangular distribution. To do this, users are required to supply an "optimistic" dictionary containing the minimum duration estimates for each activity, as well as a "pessimistic" dictionary with the maximum duration values. Subsequently, the user can execute a specified number of simulation iterations. The implementation of this process is demonstrated in the following code:

```
pessimistic = {'A':7.5,'B':3.5,'C':7.5,'D':9,'E':7.5,'F':3,'G':4.5,'H':3,'I':6,'J':4.5}
optimistic = {'A':2.5,'B':1,'C':2.5,'D':3,'E':2.5,'F':1,'G':1.5,'H':1,'I':2,'J':1.5}

p.monte_carlo_cpm_detailed(optimistic=optimistic,
                           pessimistic=pessimistic,
                           NumberIterations=1000)
```

Cheche_pm facilitates conducting a Monte Carlo analysis for buffer evaluation, enabling users to assess the impact of implementing activity buffers within the project schedule. Users are required to input an additional dictionary specifying the buffer sizes along with the corresponding activities to which these buffers should be applied. This functionality is easily accessible and can be implemented as demonstrated in the following code:

```
pessimistic = {'A':7.5,'B':3.5,'C':7.5,'D':9,'E':7.5,'F':3,'G':4.5,'H':3,'I':6,'J':4.5}
optimistic = {'A':2.5,'B':1,'C':2.5,'D':3,'E':2.5,'F':1,'G':1.5,'H':1,'I':2,'J':1.5}

buffers = {'A':0,'B':0,'C':0,'D':0,'E':0,'F':0,'G':0,'H':0,'I':0,'J':0}

p.monte_carlo_detail_buffer_analysis(optimistic=optimistic,
                                pessimistic=pessimistic,
                                buffer=buffers,
                                NumberIterations=1_000)
```

4 Conclusion

Cheche_pm stands as an all-encompassing toolkit for project scheduling, offering a powerful platform for the application of cutting-edge scheduling methodologies as well as the cultivation of innovative or refined algorithms. This capability stems from its fully Python-based infrastructure, which empowers users to effectively adapt, augment, and personalize existing models, introduce new contributions, and uphold exemplary standards. In light of the growing need for adept project management solutions and the swift advancements within this domain, combined with vigorous development and future contributions from both our team and the wider community, cheche_pm is strategically positioned to advance state-of-the-art scheduling techniques across various fields, reaching an extensive audience.

References

[1] Adedeji Bodunde Badiru. *Project management in manufacturing and high technology operations*. John Wiley & Sons, 1996.

[2] Project Management Institute. Why project talent is still in demand.

[3] Paul C Dinsmore and Terence J Cooke-Davies. *Right projects done right: from business strategy to successful project Implementation*. John Wiley & Sons, 2005.

[4] Aaron J Shenhar. One size does not fit all projects: Exploring classical contingency domains. *Management science*, 47(3):394–414, 2001.

[5] Aaron J Shenhar, Dov Dvir, Ofer Levy, and Alan C Maltz. Project success: a multidimensional strategic concept. *Long range planning*, 34(6):699–725, 2001.

[6] Aaron J Shenhar, Asher Tishler, Dov Dvir, Stanislav Lipovetsky, and Thomas Lechler. Refining the search for project success factors: a multivariate, typological approach. *R&d Management*, 32(2):111–126, 2002.

[7] Tomas Blomquist, Markus Hällgren, Andreas Nilsson, and Anders Söderholm. Project-as-practice: In search of project management research that matters. *Project management journal*, 41(1):5–16, 2010.

[8] Dov Dvir, Tzvi Raz, and Aaron J Shenhar. An empirical analysis of the relationship between project planning and project success. *International journal of project management*, 21(2):89–95, 2003.

[9] J Rodney Turner and Ralf Müller. On the nature of the project as a temporary organization. *International journal of project management*, 21(1):1–8, 2003.

[10] JA Carruthers and Albert Battersby. Advances in critical path methods. *Journal of the Operational Research Society*, 17(4):359–380, 1966.

[11] Jacek Blazewicz, Jan Karel Lenstra, and AHG Rinnooy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete applied mathematics*, 5(1):11–24, 1983.

[12] AH Russell. Cash flows in networks. *Management Science*, 16(5):357–373, 1970.

[13] Rainer Kolisch and Arno Sprecher. Psplib-a project scheduling problem library: Or software-orsep operations research software exchange program. *European journal of operational research*, 96(1):205–216, 1997.

[14] José Coelho and Mario Vanhoucke. Going to the core of hard resource-constrained project scheduling instances. *Computers & Operations Research*, 121:104976, 2020.

[15] James E Kelley. The critical-path method: resource planning and scheduling. *Industrial scheduling*, 1963.

[16] R Kolisch and S Hartmann. Experimental evaluation of heuristics for the resource-constrained project scheduling problem: An update. *European Journal of Operational Research*, 2005.

[17] Rainer Kolisch and Sönke Hartmann. *Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis*. Springer, 1999.

[18] Dimitri Golenko-Ginzburg and Aharon Gonik. Stochastic network project scheduling with non-consumable limited resources. *International journal of production economics*, 48(1):29–37, 1997.

[19] Rainer Kolisch. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90(2):320–333, 1996.

[20] Ke Y Li and Robert J Willis. An iterative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research*, 56(3):370–379, 1992.

[21] Vicente Valls, Francisco Ballestin, and Sacramento Quintanilla. Justification and rcpsp: A technique that pays. *European Journal of Operational Research*, 165(2):375–386, 2005.

[22] Thomas Selle and Jürgen Zimmermann. A bidirectional heuristic for maximizing the net present value of large-scale projects subject to limited resources. *Naval Research Logistics (NRL)*, 50(2):130–148, 2003.

[23] Francisco Ballestin, Vicente Valls, and Sacramento Quintanilla. Pre-emption in resource-constrained project scheduling. *European Journal of Operational Research*, 189(3):1136–1152, 2008.

[24] Dimitri Golenko-Ginzburg and Aharon Gonik. A heuristic for network project scheduling with random activity durations depending on the resource allocation. *International Journal of Production Economics*, 55(2):149–162, 1998.

[25] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[26] HO Hartley and AW Wortham. A statistical theory for pert critical path analysis. *Management Science*, 12(10):B–469, 1966.

[27] Mario Vanhoucke. *Project management with dynamic scheduling*. Springer, New York, USA, 2012.

[28] Erik Leuven Demeulemeester and Willy S Herroelen. *Project scheduling: a research handbook*, volume 49. Springer Science & Business Media, New York, USA, 2006.

[29] Ramon Alvarez-Valdes Olaguibel and JoseManuel Tamarit Goerlich. The project scheduling polyhedron: Dimension, facets and lifting theorems. *European Journal of Operational Research*, 67(2):204–220, 1993.

[30] Aristide Mingozzi, Vittorio Maniezzo, Salvatore Ricciardelli, and Lucio Bianco. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management science*, 44(5):714–729, 1998.

Chapter 6

Quantum discrete optimization: Hybrid quantum classical approach for strategic group formation

The content of this chapter is presented in the form of a paper titled "**Restricted Min-Norm Semi-Assignment in Group Formation: Quantum and Binary Algorithmic Integration**".

6.1 Chapter 6 introduction

Thischapter explores the quadratic semi-assignment problem QSAP, an important problem in combinatorial optimization with wide-ranging applications from manufacturing to services. It introduces a variant aimed at optimizing team formation in university settings, proving its NP-completeness. The goal is to maximize the μ-norm of assignment utilities across groups while ensuring homogeneity, introducing flexibility in utility optimization and constraints on group homogeneity and size. Methodologically, it integrates a branch-and-cut procedure for the MILP reformulation and a quantum annealing approach for the QUBO reformulation, enhancing solution quality through Magnanti-Wong Benders cuts and quantum-derived optimality bounds. Tested on both synthetic and real data, the model efficiently handles large-scale problems, demon-

strating the hybrid quantum-binary approach's effectiveness against traditional and quantum optimization methods.

Restricted Min-Norm Semi-Assignment in Group Formation: Quantum and Binary Algorithmic Integration

Stefano Nasini* Luis Fernando Pérez Armas †

IESEG School of Management, Univ. Lille, CNRS, UMR 9221 -LEM - Lille Economie Management, F-59000 Lille, France

Abstract

The quadratic semi-assignment problem holds a central position in combinatorial optimization, with diverse applications in manufacturing and service management. This paper introduces a specialized variant, motivated by the need for efficient collaborative team formation in university classrooms, and shows its NP-completeness. The model aims to maximize the μ-norm of assignment utilities within groups while adhering to predefined standards of homogeneity across different groups. Two key distinctions from the traditional quadratic semi-assignment problem include a generalized objective function allowing flexibility in utility maximization and the incorporation of between-group homogeneity and group size constraints. On the algorithmic side, we integrate a specialized branch-and-cut procedure (applied to the Mixed-Integer-Linear-Programming (MILP) reformulation of our model) and a quantum annealing procedure (applied to the Quadratic Unconstrained Binary Optimization (QUBO) reformulation of our model), which combines Magnanti-Wong Benders cuts and strong optimality bounds (generated based on the quantum annealing solutions). The methodology is tested on synthetic and real-world data, including classroom scenarios from a French business school. Our experiments cover instances with up to 16 thousand binary variables, over 6 million continuous variables, and 19 million constraints for the MILP reformulation and up to 345 thousand binary variables for the QUBO reformulation. Results demonstrate the efficacy of the proposed quantum and binary algorithmic integration, where the proposed hybrid quantum-based decomposition procedure is competitive with both hierarchical-clustering-based heuristics, CPLEX 12.9 implementation of the B&C algorithm, and state-of-the-art Benders decomposition (implemented by the CPLEX callable library).

Keywords: Assignment problems, Group formation, Quantum annealing, Decomposition algorithms

*s.nasini@ieseg.fr

†l.perez@ieseg.fr

1 Introduction

The Quadratic Semi-Assignment Problem (QSAP) involves assigning objects (facilities) from one set to another (locations) while taking into account the distance between each pair of locations and the transport flow between any pair of facilities. This problem has been proved to be NP-hard and holds a central position in the domain of combinatorial optimization (Malucelli and Pretolani, 1995; Billionnet and Elloumi, 2001). As noted by Silva et al. (2021), despite being easy to define, QSAPs are very difficult to solve in practice, even for instances of small size, giving rise to a long stream of contributions focusing on their heuristic and metaheuristic solvability.

These methodologies have become particularly important given the large number of applications of QSAPs in different manufacturing and service management contexts. Common applications include the assignment of clients to processors, where increased delays are caused by multiple clients competing for a single processor (Magirou and Milis, 1989), and task-machine assignments with constraints imposing desired properties for groups of tasks (Gallo and Simeone, 1991). Further applications deal with hub location (Saito et al., 2009), as well as with specialized variants of correlation clustering, which is of particular importance in group formation (Demaine et al., 2006).

In operations management, group formation problems have been at the center of numerous contributions, since the work of Cool and Schendel (1987), who study the performance implications of strategic groups for the industrial organization. Groups also play a crucial role in project management and education.

Motivated by the need to form efficient collaborative teams within university classrooms, we present in this paper a Restricted Min-Norm Semi-Assignment Problem (R-MN-SAP), a specialized variant of the QSAP. In this variant, the primary objective is to maximize the μ-norm of assignment utilities within groups, while adhering to predefined standards of homogeneity across various groups. The distinctive elements of the R-MN-SAP formulation in comparison to the well-known QSAP are twofold. Firstly, focusing on the μ-norm of within-group utility, we establish a selection criterion for the multi-objective problem of maximizing each group utilities, which encodes the traditional quadratic forms of the QSAP objective function. This entails a generalized objective function form of the QSAP, as $\mu = 1$ implies the maximization of the sum of utilities (in line with the traditional QSAP), and $\mu = \infty$ implies the maximization of the minimum utility. Secondly, the inclusion of between-group homogeneity constraints and group size constraints constitutes an important distinction with respect to the traditional QSAP formulation, which allows addressing real-world applications where the group formation must satisfy specific standards required by institutional or legal regulations. These variations reflect not only the institutional requirements of our analyzed university classroom application, but also the modeling orientation in the group formation literature. For instance, Carrell et al. (2013) adopt a maximization of the minimum to design students groups that improve the academic performance of the lowest ability students; Pycia (2012) studies the role of pairwise preferences in group formation and coalition stability. Although cooperative games have been frequently adopted to study coalition stability among business partners and competing firms (Basso et al., 2021; Jouida et al., 2017), the benefit of the R-MN-SAP is quantifiable in institutional contexts in which an administrative authority (i.e., school and academic center) is entitled to take centralized decisions on the group design.

On the algorithmic side, the R-MN-SAP is approached by the integration of heuristic and exact solution strategies within a specialized branch-and-cut procedure, which combines Magnanti-Wong Benders cuts (Magnanti and Wong, 1981; Papadakos, 2008) and strong optimality bounds. Specifically, our solution methodology considers the traditional MILP reformulations of the R-MN-SAP and applies a specialized Benders decomposition approach, which allows eliminating a quadratic number of decision variables, by appending optimality cuts at the integer nodes of the branch-and-bound tree.

We enrich the cut generation procedure for the construction of Magnanti-Wong core points and strong optimality bounds, using high performing heuristic solutions. To do so, we exploit the fact that the R-MN-SAP can be suitably re-formulated as a QUBO problem, which can in turn be solved using the Quantum Annealing (QA) algorithm. This is a quantum metaheuristic specifically designed to solve QUBO problems utilizing the principles of quantum mechanics (Kadowaki and Nishimori, 1998; Albash and Lidar, 2018).

While the specificity of the QA algorithm is to attain almost optimal solutions in a few seconds, using this high performing solution approach to enrich our cut generation allows attaining an optimality signature when running our specialized branch-and-cut algorithm. As a result, our quantum and binary algorithmic integration relies on tackling the QUBO reformulation of the R-MN-SAP by the QA heuristic and the MILP reformulation by a specialized branch-and-cut algorithm, which benefits from QA solution to generate optimality bounds and Magnanti-Wong Benders cuts.

We test our methodology using synthetic and real-world data. The latter consists of two classrooms of 21 and 49 students from a French business school. Synthetic data are constructed to reflect the working group formation of the bi-annual meetings of professors from the same business school, covering instances with up to 16 thousand binary variables, over 6 million continuous variables, and 19 million constraints for the MILP reformulation and up to 345 thousand binary variables for the QUBO reformulation.

For comparative purposes, a traditional hierarchical-clustering-based (HC-based) algorithm is used to show that these results cannot be obtained if our cut generation is established using an alternative heuristic method. In fact, the QA approach is broadly capable of attaining almost optimal solutions in a few seconds, which have, however, no optimality signature. Using the QA solution to generate Magnanti-Wong core points, lower bounds, and initial Benders cut allows the branch-and-cut to prune a large number of fractional and integer nodes. Hence, our quantum and binary-algorithmic integration is competitive with both HC-based heuristics, CPLEX 12.9 implementation of the B&C algorithm, and state-of-the-art Benders decomposition (implemented by the CPLEX callable library). For the large-scale instances, we show that our approach allows attaining solutions that could be achieved neither by the QA algorithm applied to the QUBO reformulation of the R-MN-SAP nor by the Benders decomposition algorithm, applied to the corresponding MILP reformulation.

The rest of the contribution is organized as follows. Section 2 provides a literature review. Section 3 presents our modeling design for the R-MN-SAP. Section 4 presents algorithmic strategies to approach the R-MN-SAP, with specific focus on the specialized branch-and-cut algorithm. Section 5 presents the QA algorithm for the R-MN-SAP. Section 6 presents an illustrative example of the proposed methodology. The empirical and computational analysis is provided in Section 7. Finally, Section 8 concludes the paper. We collect all mathematical proofs in the online supplementary appendix.

2 Related literature

To elucidate the positioning of our paper, three streams of literature are discussed in this section, highlighting the relationship with the introduced R-MN-SAP.

Assignment and quadratic assignment problems. The QSAP was introduced by Greenberg (1969) as a variant of the original Quadratic Assignment Problem (QAP) of Koopmans and Beckmann (1957) and Lawler (1963) (which in turn generalizes the Linear Assignment Problem (LAP)). While the QAP consists of assigning n facilities to n locations, in such a way that each facility can be located in one site and each site can hold exactly one facility, the QSAP relaxes the latter

constraint, allowing multiple facilities to be assigned to the same location. This entails a direct applicability of the QSAP to group formation, where each facility represents an individual and each location represents a group. Given the well-documented real-world applications of the QSAP, a long stream of contributions in the discrete mathematics and operations research literature has a focus on providing efficient solution methods, for specific variants of the QSAP. While being NP-hard in general, certain polynomially solvable cases and lower bounds for the QSAP have been studied in in the operations research literature, such as the works of Malucelli and Pretolani (1994, 1995) and Malucelli (1996). Another example is the work of Drwal (2014), who shows that the maximization of the worst-case group m-partitioning with no constraints can be solved in polynomial time. A closely related version of QSAPs is the quadratic multi-knapsack (QMKP) problem (Bergman, 2019; Olivier et al., 2021), which include group size constraints to the traditional QSAP. Our contribution to this stream of literature focuses closely on extending the QSAP to a more general class of group formation problems, when pairwise preferences, group utility aggregation, and institutional regulations are taken into account.

Specialized decomposition procedures. Decomposition algorithms have been largely used to address multiple classes of uncapacitated (Fischetti et al., 2017) and capacitated facility location problems (Wentges, 1996; Castro et al., 2017), by nondifferentiable cutting-plane approaches. Facility location problems and group formation problems are closely related and aim to optimally group entities based on specific criteria (Neshati et al., 2014). In fact, in facility location problems, facilities need to be strategically placed to minimize costs, while in group formation problems, individuals or teams need to be grouped to achieve specific goals. The use of a specialized decomposition algorithm for the R-MN-SAP mirrors the idea of projecting the MILP reformulation of the R-MN-SAP into the binary variable space, in line with Fischetti et al. (2017), Wentges (1996) and Castro et al. (2017). The specificity of our decomposition procedure for the R-MN-SAP relies upon the use of high performing heuristic solutions (obtained by approaching the QUBO reformulation of the R-MN-SAP by the QA heuristic) for the construction of Magnanti-Wong core points and strong optimality bounds (Magnanti and Wong, 1981; Papadakos, 2008).

Quadratic unconstrained binary optimization. Multiple optimization problems have been successfully mapped into the QUBO formulation, such as the job shop scheduling problem (Carugno et al., 2022), the SAT and the MaxSAT (Bian et al., 2020), traffic flow optimization problems (Neukart et al., 2017), finding optimal trajectories for air traffic management (Stollenwerk et al., 2019), among others (Glover (a) et al., 2022; Glover (b) et al., 2022). The idea of QA for QUBO problems mirrors the one of simulated annealing (SA) (Kirkpatrick et al., 1983; Koopmans and Beckmann, 1957; Burkard and Rendl, 1984) and can be introduced by considering a continuous-time stochastic process operating on a set of binary vectors (the feasible region of the QUBO problem). This stochastic process constitutes a quantum metaheuristic that relies upon the adiabatic theorem of quantum mechanics and the properties of the time-dependent Schrödinger equation (Kadowaki and Nishimori, 1998; Albash and Lidar, 2018). As such, the QA approach can only be run on specialized quantum computers, which are remotely accessible by means of dedicated programming libraries.[1] In our contribution, we map the R-MN-SAP into a QUBO formulation with a view to applying the QA approach to generate Magnanti-Wong core points and strong optimality bounds.

[1]The benefit of quantum computing in optimization is not limited to the QUBO problem. Nannicini (2022) extends the range of applicability by proposing quantum subroutines for the simplex method which avoids classical computation of the basis inverse. Bova et al. (2023) investigate the economic implications of solving optimization problems by quantum computing in a duopoly, where a quantum computing company competes against a classical computing one. For an overview of the synergy between operations research and quantum computing, see Parekh (2023).

While the specificity of the QA algorithm is to attain almost optimal solutions in a few seconds, our contribution allows exploiting this high performing solution approach to enrich our cut generation procedure.

3 Model

We consider a university classroom with n students and m group projects to be carried out, and introduce the notations that will be used throughout the paper.

Sets:

$\mathcal{N}$	the set of students (with $\|\mathcal{N}\| = n$);
$\mathcal{M}$	the set of groups (with $\|\mathcal{M}\| = m$);
$\mathcal{F}$	a set of students features (with $\|\mathcal{F}\| = F$).

Exogenous parameters:

$\alpha_{i,j} \in [-1, 1]$	interpersonal affinity/aversion that student $i \in \mathcal{N}$ has for student $j \in \mathcal{N}$;
$\beta_{i,m} \in [-1, 1]$	the degree of interest of student $i \in \mathcal{N}$ for project $m \in \mathcal{M}$;
$\gamma_i^f \in \mathbb{R}$	the value of feature $f \in \mathcal{F}$ in student $i \in \mathcal{N}$;
$\tau^{\max} \in \{0, \ldots, n\}$	maximum number of students per group;
$\tau^{\min} \in \{0, \ldots, n\}$	minimum number of students per group;
$\theta_f \in \mathbb{R}$	maximum between-group heterogeneity level, for each $f \in \mathcal{F}$.

Endogenous model decisions:

$\lambda : \mathcal{N} \to \mathcal{M}$	an assignment function;
$x_{i,c} \in \{0, 1\}$	the binary indicator of whether student i is assigned to group c.

We define $A = (\alpha_{i,j})_{i \in \mathcal{N}, j \in \mathcal{N}} \in \mathbb{R}^{n \times n}$ and $B = (\beta_{i,c})_{i \in \mathcal{N}, c \in \mathcal{M}} \in \mathbb{R}^{n \times m}$. The QSAP is that of assigning students to groups so that the μ-norm of groups dissatisfaction is minimized. Disutility functions are defined in line with the empirical evidence provided by Axelrod et al. (1995), where the disutility of an individual for joining a group decreases with the size of the group and increases with the presence of rivals in the group. These quantify the within-group affinity and preference for specified group projects:

$$\delta_c(\lambda) = \kappa - \sum_{(i,j) \in \mathcal{E}} \alpha_{i,j} d(\lambda(i), \lambda(j)) - \sum_{i \in \lambda^{-1}(c)} \beta_{i,c},$$

where $\kappa \geq \alpha_{\max} n^2 + \beta_{\max,c} n$ is an exogenous constant used to guarantee the non-negativity of the disutility functions, and $d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ is an indicator function (i.e., $d(i, j) = 1$ if $i = j$ and $d(i, j) = 0$ if $i \neq j$). The minimization of the total disutility constitutes a QSAP (Greenberg, 1969; Saito et al., 2009). Intuitively, this minimization entails that a pair of students (i, j) assigned to the same group $c \in \mathcal{M}$ must not only have a high mutual preference $\alpha_{i,j}$ and $\alpha_{j,i}$, but also a high structural similarity (Pollalis and Mavrommatis, 2009), as they should maintain a good relationship with the same collection of other students in group c. In the social network literature, two nodes are regarded as structurally equivalent if they have precisely the same set of relations across all other nodes in the network. Consistently, structural similarity refers to the similarity between the set of relations of two nodes, which attains the highest value when two nodes are structurally equivalent.[2]

[2]The idea of structural similarity will be used in Section 7 to construct a heuristic method for the solution of the R-MN-SAP, in line with the approach of Pollalis and Mavrommatis (2009).

Proposition 1 (structurally equivalent students)**.** *Let $i, j \in \mathcal{N}$ be two structurally equivalent students and consider an assignment solution $\lambda(h)$, for all $h \in \mathcal{N} \setminus \{i, j\}$. We claim that if*

$$\alpha_{i,j} + \alpha_{j,i} \geq 1 + \beta_{i,c} - \sum_{h \in \mathcal{N} \setminus \{i,j\}} \mathbb{1}\{\lambda(h) \neq c\}(\alpha_{i,h} + \alpha_{h,i}),$$

then $\lambda(i) = \lambda(j) = c$, for some $c \in \mathcal{M}$.

The R-MN-SAP is a specialized variant of the QSAP problem, defined as the minimization of the μ-norm of groups disutilities:

$$\sigma(\mu) = \left\{ \min_{\boldsymbol{\lambda}} \quad \left(\sum_{c \in \mathcal{M}} (\delta_c(\lambda))^{\mu} \right)^{1/\mu} \qquad \text{subject to } (\lambda(1), \ldots, \lambda(n)) \in \Lambda, \right. \tag{1}$$

where Λ is a set of all assignment functions satisfying the following constraints:

(i) **Group size constraints**. For all $i \in \mathcal{N}$ and $c \in \mathcal{M}$, $\tau_{\min} \leq |\lambda(i)^{-1}(c)| \leq \tau_{\max}$.

(ii) **Homogeneity constraints**. For all $i \in \mathcal{N}$ and $c,\ c' \in \mathcal{M}$, $|\sum_{i \in \lambda^{-1}(c)} \gamma_i^f - \sum_{i \in \lambda^{-1}(c')} \gamma_i^f| \leq \theta_f$.

The inclusion of (i) group size constraints and (ii) between-group homogeneity constraints set up an important departure from the traditional QSAP formulation. This allows addressing real-world applications where the group formation is required to satisfy specific standards imposed by institutional or legal regulations. For instance, in the empirical application studied in Subsection 7.1, the academic institution actively promotes diversity within the working groups, which entails between-group homogeneity.[3]

Remark 1 (**Min-sum and min-max generalization**)**.** *Problem* (1) *covers a class of problems that ranges from the min-sum semi-assignment problem (when $\mu = 1$) to the min-max semi-assignment problem (when $\mu = \infty$):*

$$\sigma(1) \quad = \left\{ \begin{array}{ll} \min\limits_{\boldsymbol{\lambda}} & \sum\limits_{c \in \mathcal{M}} \left(\kappa - \sum\limits_{(i,j) \in \mathcal{E}} \alpha_{i,j} d(\lambda(i), \lambda(j)) - \sum\limits_{i \in \lambda^{-1}(c)} \beta_{i,c} \right) \\ \textit{subj. to} & (\lambda(1), \ldots, \lambda(n)) \in \Lambda, \end{array} \right. \tag{2}$$

$$\sigma(\infty) \quad = \left\{ \begin{array}{ll} \min\limits_{\boldsymbol{\lambda}} & \max\limits_{c \in \mathcal{M}} \left(\kappa - \sum\limits_{(i,j) \in \mathcal{E}} \alpha_{i,j} d(\lambda(i), \lambda(j)) - \sum\limits_{i \in \lambda^{-1}(c)} \beta_{i,c} \right) \\ \textit{subj. to} & (\lambda(1), \ldots, \lambda(n)) \in \Lambda. \end{array} \right. \tag{3}$$

It is important to note that R-MN-SAP and QSAP are not nested, as are R-MN-SAP and LAP and R-MN-SAP and QMKP. However, they can address closely related problems involving agent-task assignment and group formation, as established below.

Remark 2 (**Reduction to the LAP**)**.** *When $n = m$, $\tau_{\min} = \tau_{\max} = 1$, and $\theta_f \geq \gamma^f_{\max}$, the min-sum problem* (2) *reduces to a LAP consisting of assigning n agents to n tasks with costs $-\beta_{i,c}$ (Burkard et al., 2012).*

Remark 3 (**Reduction to the QSAP**)**.** *When $\tau_{\min} = 0$, $\tau_{\max} = n$ and $\theta_f \geq \gamma^f_{\max}$, the min-sum problem* (2) *reduces to a QSAP (Malucelli and Pretolani, 1995; Billionnet and Elloumi, 2001).*

[3]The performance implications of diversity within working groups are supported by Milliken and Martins (1996).

Remark 4 (**Reduction to the QMKP**). *When $\tau_{\min} = 0$ and $\theta_f \geq \gamma^f_{\max}$, the min-sum problem* (2) *reduces to a QMKP with homogeneous weights (Bergman, 2019; Olivier et al., 2021).*

We study hereafter the complexity class of the R-MN-SAP.

Proposition 2 (NP completeness). *Let μ-R-MN-SAP be the recognition version of R-MN-SAP for a specific $\mu \in \mathbb{R}_+$, which asks whether there exists a m groups that solves* (1) *with an objective function larger than or equal to a given threshold. We claim that for each $\mu \in \mathbb{R}_+$, μ-R-MN-SAP is NP-complete.*

While the complexity class of the R-MN-SAP is deduced for the general case of $\mu \in \mathbb{R}_+$, the rest of the paper focuses specifically on $\mu = 1$ (min-sum problem (2)) and $\mu = \infty$ (min-max problem (3)). In these two cases, MILP and QUBO reformulations can be readily obtained. For all the other values of μ, a relaxation of problem (1) can be approached by piecewise linear approximations, by defining a collection $\mathcal{T}$ of interpolation seeds with values ℓ_t, for $t \in \mathcal{T}$, obtaining:

$$\sigma^R(\mu) = \begin{cases} \min\limits_{\boldsymbol{\lambda}} & \phi \\ \text{s. to} & \phi \geq \left(\sum_{c\in\mathcal{M}} (\ell_t)^\mu\right)^{1/\mu} + \sum_{c\in\mathcal{M}} (\ell_t)^{\mu-1} \left(m\,(\ell_t)^{\mu-1}\right)^{1/\mu} (\delta_c(\lambda) - \ell_t) \quad t \in \mathcal{T} \\ & (\lambda(1), \ldots, \lambda(n)) \in \Lambda. \end{cases} \tag{4}$$

Hence, $\sigma^R(\mu) \leq \sigma(\mu)$, for all $\mu \in \mathbb{R}_+$.

4 Solution strategies

Building on the complexity class of the R-MN-SAP, this section is dedicated to the construction of a specialized branch-and-cut procedure to tackle large instances of the min-sum problem (2) and the min-max problem (3), based on a MILP reformulation. Because the optimal solutions of both problems is invariant with respect to κ, this parameter is omitted throughout this section.

4.1 Binary programming characterization

Problems (2) and (3) can be formulated as the following quadratic programming problems:

$$\begin{cases} \max & \sum\limits_{c\in\mathcal{M}} \eta_c & & \text{(5a)} \\ \text{subj. to} & \eta_c \leq \sum\limits_{i\in\mathcal{N}} \sum\limits_{j\in\mathcal{N}} \alpha_{i,j} x_{i,c} x_{j,c} + \sum\limits_{i\in\mathcal{N}} \beta_{i,c} x_{i,c} & c \in \mathcal{M} & \text{(5b)} \\ & x_{i,c} \text{ satisfying constraints (7a)-(7e)}, & & \text{(5c)} \end{cases}$$

and

$$\begin{cases} \max & \eta & & \text{(6a)} \\ \text{subj. to} & \eta \leq \sum\limits_{i>j} \alpha_{i,j} x_{i,c} x_{j,c} + \sum\limits_{i\in\mathcal{N}} \beta_{i,c} x_{i,c} & c \in \mathcal{M} & \text{(6b)} \\ & x_{i,c} \text{ satisfying constraints (7a)-(7e)}, & & \text{(6c)} \end{cases}$$

respectively, where

$$\begin{cases} \sum_{c\in\mathcal{M}} x_{i,c} = 1 & i \in \mathcal{N} \quad (7a) \\ \sum_{i\in\mathcal{N}} x_{i,c} \geq \tau^{\min} & c \in \mathcal{M} \quad (7b) \\ \sum_{i\in\mathcal{N}} x_{i,c} \leq \tau^{\max} & c \in \mathcal{M} \quad (7c) \\ \sum_{i\in\mathcal{N}} \gamma_i^f (x_{i,c} - x_{i,c'}) \leq \theta_f & f \in \mathcal{F},\ c, c' \in \mathcal{M} \quad (7d) \\ x_{i,c} \ \text{ binary} & i \in \mathcal{N},\ c \in \mathcal{M}. \quad (7e) \end{cases}$$

Constraints (7a) establish that each student must belong to exactly one group; constraints (7b)-(7c) establish a minimum and maximum numbers of students per group; constraints (7d) enforce a maximum between-group heterogeneity, by preventing extreme variations of the aggregate group levels with respect to features $f \in \mathcal{F}$ between different groups.

The quadratic terms in the presented models can be linearized using common techniques (Glover and Woolsey, 1974; Sherali and Adams, 1998). Hence, linear programming reformulations of problems (5) and (6) are obtained by introducing variables $w_{i,j}^c$ to encode the bi-linear term $x_{i,c}x_{j,c}$. We obtain

$$\begin{cases} w_{i,j}^c \leq x_{i,c} & i,j \in \mathcal{N}\ (i \neq j),\ c \in \mathcal{M} \quad (8a) \\ w_{i,j}^c \leq x_{j,c} & i,j \in \mathcal{N}\ (i \neq j),\ c \in \mathcal{M} \quad (8b) \\ w_{i,j}^c \geq x_{i,c} + x_{j,c} - 1 & i,j \in \mathcal{N}\ (i \neq j),\ c \in \mathcal{M} \quad (8c) \\ w_{i,i}^c = x_{i,c} & i \in \mathcal{N},\ c \in \mathcal{M} \quad (8d) \\ w_{i,j}^c \geq 0 & i,j \in \mathcal{N},\ c \in \mathcal{M}, \quad (8e) \end{cases}$$

with

$$\sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} \alpha_{i,j} x_{i,c} x_{j,c} + \sum_{i\in\mathcal{N}} \beta_{i,c} x_{i,c} = \sum_{i>j} \alpha_{i,j} w_{i,j}^c + \sum_{i\in\mathcal{N}} \beta_{i,c} x_{i,c}.$$

Hence, the linear reformulation of (5) and (6) requires $m(n^2 - n)$ continious variables (after pre-reduction using the equality constraints), nm binary variables, and $2m + m(m-1)F$ linear inequality constraints. For the sake of exhaustiveness, a MILP reformulation of the relaxed problem (4) is provided in Appendix D.

4.2 Benders decomposition

Problems (5) and (6), as described in Section 4.1, are good candidates for the application of the Benders decomposition approach (Benders, 2005; Magnanti and Wong, 1981; Rei et al., 2009; Wentges, 1996). In fact, once a decision is made for the binary x-variables, the remaining problem admits a closed-form solution with respect to the w-variables. Therefore, the problem can be projected onto the x-variables space yielding

$$\begin{cases} \max \quad \sum_{c\in\mathcal{M}} \eta_c & (9a) \\ \text{subj. to} \quad \eta_c \leq \sum_{(i,j)\in\mathcal{E}} Q(x_{i,c}, x_{j,c}) & c \in \mathcal{M} \quad (9b) \\ \qquad\qquad x_{i,c} \text{ satisfying constraints (7a)-(7e)}, & (9c) \end{cases}$$

for problem (5), and

$$\begin{cases} \max & \eta & & (10a) \\ \text{subj. to} & \eta \leq \sum_{(i,j)\in\mathcal{E}} Q(x_{i,c}, x_{j,c}) & c \in \mathcal{M} & (10b) \\ & x_{i,c} \text{ satisfying constraints (7a)-(7e)}, & & (10c) \end{cases}$$

for problem (6), where for each $i, j \in \mathcal{N}$ and $c \in \mathcal{M}$,

$$Q(x_{i,c}, x_{j,c}) = \begin{cases} \max & \alpha_{i,j} w^c_{i,j} & & (11a) \\ \text{subj. to} & w^c_{i,j} \leq x_{i,c} & [\overline{\zeta}^c_{i,j}] & (11b) \\ & w^c_{i,j} \leq x_{j,c} & [\underline{\zeta}^c_{i,j}] & (11c) \\ & w^c_{i,j} \geq x_{i,c} + x_{j,c} - 1 & [\zeta^c_{i,j}] & (11d) \\ & w^c_{i,j} \geq 0, & & (11e) \end{cases}$$

with $\overline{\zeta}^c_{i,j}$, $\underline{\zeta}^c_{i,j}$ and $\zeta^c_{i,j}$ being Lagrangian multipliers associated to the equivalent dual formulation. Hence, the dual Benders subproblem can be expressed as

$$Q(x_{i,c}, x_{j,c}) = \begin{cases} \min & x_{i,c}\overline{\zeta}^c_{i,j} + x_{j,c}\underline{\zeta}^c_{i,j} + (1 - x_{i,c} - x_{j,c})\zeta^c_{i,j} & (12a) \\ \text{subj. to} & \overline{\zeta}^c_{i,j} + \underline{\zeta}^c_{i,j} - \zeta^c_{i,j} \geq \alpha_{i,j} & (12b) \\ & \overline{\zeta}^c_{i,j}, \underline{\zeta}^c_{i,j}, \zeta^c_{i,j} \geq 0. & (12c) \end{cases}$$

Because $Q(x_{i,c}, x_{j,c})$ is a concave piecewise linear function, problems (9) and (10) can be approached by some nondifferentiable cutting-plane procedures. Benders decomposition can be seen as a particular implementation of such a procedure, where $Q(x_{i,c}, x_{j,c})$ is approximated from above by cutting planes. These planes are obtained by evaluating $Q(x_{i,c}, x_{j,c})$ at some particular $x_{i,c}$ and $x_{j,c}$ values, i.e., solving the (Benders) subproblem induced by those values. The new cuts replace $Q(x_{i,c}, x_{j,c})$ and are sequentially added to problems (9) and (10), leading to the following updated (Benders) master problem:

$$\begin{cases} \max & \sum_{c\in\mathcal{M}} \eta_c & & (13a) \\ \text{s. to } \eta_c \leq & \sum_{(i,j)\in\mathcal{E}} \left(x_{i,c}\overline{\zeta}^c_{i,j}(v) + x_{j,c}\underline{\zeta}^c_{i,j}(v) + (1 - x_{i,c} - x_{j,c})\zeta^c_{i,j}(v) \right) + \sum_{i\in\mathcal{N}} \beta_{i,c} x_{i,c} & & (13b) \\ & x_{i,c} \text{ satisfying constraints (7a)-(7e)}, & & (13c) \end{cases}$$

and

$$\begin{cases} \max & \eta & & (14a) \\ \text{s. to } \eta \leq & \sum_{(i,j)\in\mathcal{E}} \left(x_{i,c}\overline{\zeta}^c_{i,j}(v) + x_{j,c}\underline{\zeta}^c_{i,j}(v) + (1 - x_{i,c} - x_{j,c})\zeta^c_{i,j}(v) \right) + \sum_{i\in\mathcal{N}} \beta_{i,c} x_{i,c} & c \in \mathcal{M} & (14b) \\ & x_{i,c} \text{ satisfying constraints (7a)-(7e)}, & & (14c) \end{cases}$$

where the v index in the dual variables appearing in constraints (13b) and (14b) corresponds to the ordering of Benders cuts generated along the cutting-plane algorithm.

Benders master and subproblem provide, respectively, lower and upper bounds to the optimal solution. Such a cutting-plane algorithm is iterated until the gap between the lower and upper bound is small enough or zero. The following proposition allows establishing a closed-form cut generation.

Proposition 3 (Dual subproblem)**.** *Problem* (12) *admits an infinite number of optimal solutions, from which we characterize the following specific solution:*

$$\begin{array}{llll}
\textit{if } x_{i,c}+x_{j,c}=1, & \textit{then} & \overline{\zeta}^c_{i,j}=(\alpha_{i,j})^+(1-x_{i,c}), & \underline{\zeta}^c_{i,j}=(\alpha_{i,j})^+(1-x_{j,c}), \quad \zeta^c_{i,j}=0,\\
\textit{if } x_{j,c}+x_{j,c}=2, & \textit{then} & \overline{\zeta}^c_{i}=0, & \underline{\zeta}^c_{j}=0, \quad \zeta^c_{i,j}=(\alpha_{i,j})^+,\\
\textit{if } x_{i,c}+x_{j,c}=0, & \textit{then} & \overline{\zeta}^c_{i}=\dfrac{(\alpha_{i,j})^+}{2}, & \underline{\zeta}^c_{j}=\dfrac{(\alpha_{i,j})^+}{2}, \quad \zeta^c_{i,j}=0,
\end{array}$$

where $(\alpha_{i,j})^+=\max\{0,\alpha_{i,j}\}$.

Corollary 1 (Closed-form Benders cut)**.** *Let* $x_{i,c}(v)$ *be the solution of the Master problem at iteration* v*, for each* $i\in\mathcal{N}$ *and* $c\in\mathcal{M}$*. Constraints* (13b) *and* (14b) *reduce to*

$$\left\{\begin{array}{lll}\eta_c & \leq \displaystyle\sum_{i\in\mathcal{N}} x_{i,c}(\tilde{\alpha}^0_{i,c}(v)+\tilde{\alpha}^1_{i,c}(v))\\ \eta & \leq \displaystyle\sum_{i\in\mathcal{N}} x_{i,c}(\tilde{\alpha}^0_{i,c}(v)+\tilde{\alpha}^1_{i,c}(v)),\end{array}\right. \tag{15}$$

where

$$\tilde{\alpha}^0_{i,c}(v)=\sum_{j\in\mathcal{N}}\overline{\zeta}^c_{i,j}(v) \quad \textit{and} \quad \tilde{\alpha}^1_{i,c}(v)=\sum_{i\in\mathcal{N}}\underline{\zeta}^c_{i,j}(v).$$

The Benders subproblem (11) is degenerate, so that one can generate a whole set of different cuts in the form of (15). Magnanti and Wong (1981) formulate a problem that generates the strongest cut possible, where the strengths of two cuts corresponding to v and v' are compared by establishing that v dominates v' if and only if

$$\sum_{i\in\mathcal{N}} x_{i,c}(\tilde{\alpha}^0_{i,c}(v)+\tilde{\alpha}^1_{i,c}(v)) \geq \sum_{i\in\mathcal{N}} x_{i,c}(\tilde{\alpha}^0_{i,c}(v')+\tilde{\alpha}^1_{i,c}(v')) \qquad \forall\, \boldsymbol{x}\in\mathcal{X},$$

where $\mathcal{X}$ is the set of feasible solutions defined by constraints (7a)-(7e). Let $\texttt{CH}(\mathcal{X})$ be the convex hull of $\mathcal{X}$. A cut is Pareto-optimal if it is not dominated by any other cut. The cut generation procedure proposed by Magnanti and Wong (1981) depends on the Benders subproblem and on a Benders master problem core point. A core point is a point in the relative interior of $\texttt{CH}(\mathcal{X})$ (i.e., the largest subspace for which $\texttt{CH}(\mathcal{X})$ is full dimensional). The following proposition defines the core-points for problems (5) and (6).

Proposition 4 (Magnanti-Wong cuts)**.** *Let us assume* $\tau_{\min}\leq n/m\leq\tau_{\max}$*. For any point* $\boldsymbol{x}_0\in\mathcal{X}$*, a core point can be constructed as* $\tilde{\boldsymbol{x}}=\iota\boldsymbol{x}_0+(1-\iota)\boldsymbol{y}$*, for any* $\iota\in(0,1)$*, where* $\boldsymbol{y}$ *is defined as*

$$y_{i,c}=\frac{1}{n}\left(\tilde{\iota}\tau_{\max}+(1-\tilde{\iota})\tau_{\min}\right), \qquad \textit{and} \qquad \tilde{\iota}=\frac{\dfrac{n}{m}-\tau_{\min}}{\tau_{\max}-\tau_{\min}}.$$

Therefore, the point $\tilde{\boldsymbol{x}}=\iota\boldsymbol{x}_0+(1-\iota)\boldsymbol{y}$ is in the relative interior of $\texttt{CH}(\mathcal{X})$. Following Papadakos (2008), the Pareto-optimal cuts proposed by Magnanti and Wong (1981) are obtained by solving:

$$\tilde{Q}(x_{i,c},x_{j,c},\tilde{x}_{i,c},\tilde{x}_{j,c})=\left\{\begin{array}{rll}
\min & \tilde{x}_{i,c}\overline{\zeta}^c_{i,j}+\tilde{x}_{j,c}\underline{\zeta}^c_{i,j}+(1-\tilde{x}_{i,c}-\tilde{x}_{j,c})\zeta^c_{i,j} & (16a)\\
\text{subj. to} & \overline{\zeta}^c_{i,j}+\underline{\zeta}^c_{i,j}-\zeta^c_{i,j}\geq\alpha_{i,j} & (16b)\\
& x_{i,c}\overline{\zeta}^c_{i,j}+x_{j,c}\underline{\zeta}^c_{i,j}+(1-x_{i,c}-x_{j,c})\zeta^c_{i,j}=\alpha_{i,j}x_{i,c}x_{j,c} & (16c)\\
& \overline{\zeta}^c_{i,j},\underline{\zeta}^c_{i,j},\zeta^c_{i,j}\geq 0, & (16d)
\end{array}\right.$$

where $\tilde{x}_{i,c}$ is the element of $\tilde{\boldsymbol{x}}$, corresponding to $i\in\mathcal{N}$ and $c\in\mathcal{M}$.

Proposition 5 (Magnanti-Wong cut)**.** *The optimal solutions of problem* (16) *are the following:*

if $x_{i,c} = x_{j,c} = 1$ *and* $\tilde{x}_{i,c} \leq \tilde{x}_{j,c}$ *then:*	$\overline{\zeta}^c_{i,j} = (\alpha_{i,j})^+,$	$\underline{\zeta}^c_{i,j} = 0,$	$\zeta^c_{i,j} = 0,$
if $x_{i,c} = x_{j,c} = 1$ *and* $\tilde{x}_{j,c} \leq \tilde{x}_{i,c}$ *then:*	$\overline{\zeta}^c_{i,j} = 0,$	$\underline{\zeta}^c_{i,j} = (\alpha_{i,j})^+,$	$\zeta^c_{i,j} = 0,$
if $x_{i,c} = 1,\ x_{j,c} = 0$ *and* $\tilde{x}_{i,c} \leq 1 - \tilde{x}_{i,c} - \tilde{x}_{j,c}$ *then:*	$\overline{\zeta}^c_{i,j} = (\alpha_{i,j})^+,$	$\underline{\zeta}^c_{i,j} = 0,$	$\zeta^c_{i,j} = 0,$
if $x_{i,c} = 0,\ x_{j,c} = 1$ *and* $\tilde{x}_{j,c} \leq 1 - \tilde{x}_{i,c} - \tilde{x}_{j,c}$ *then:*	$\overline{\zeta}^c_{i,j} = 0,$	$\underline{\zeta}^c_{i,j} = (\alpha_{i,j})^+,$	$\zeta^c_{i,j} = 0.$
if $x_{i,c} = 1,\ x_{j,c} = 0$ *and* $\tilde{x}_{i,c} > 1 - \tilde{x}_{i,c} - \tilde{x}_{j,c}$ *then:*	$\overline{\zeta}^c_{i,j} = 0,$	$\underline{\zeta}^c_{i,j} = (-\alpha_{i,j})^+,$	$\zeta^c_{i,j} = 0,$
if $x_{i,c} = 0,\ x_{j,c} = 1$ *and* $\tilde{x}_{j,c} > 1 - \tilde{x}_{i,c} - \tilde{x}_{j,c}$ *then:*	$\overline{\zeta}^c_{i,j} = 0,$	$\underline{\zeta}^c_{i,j} = 0,$	$\zeta^c_{i,j} = (-\alpha_{i,j})^+.$

Based on Propositions 3 and 5, both traditional and Pareto efficient Benders cut can be computed in closed–form, which allows removing variables $w^c_{i,j}$ from the linear reformulation (8a)-(8e), with no computational effort to solve the Benders subproblem.

5 Quantum annealing for R-MN-SAP

Burkard and Rendl (1984) applied the SA algorithm to solve the QAP introduced by Koopmans and Beckmann (1957), revealing a competitive computational performance, where suboptimal solutions differing 1-2% from the best known solutions are obtained in a few seconds.

Addressing problems (2) and (3) by the SA algorithm involves creating a discrete-time Markov chain operating within the feasible solution space $\mathcal{X}$, defined by the constraints (7a)-(7e). Ideally, the transition probabilities of this Markov chain must be designed to ensure that its limiting distribution converges toward the optimal solution of problems (2) and (3). In a similar way, the idea of the QA algorithm for problems (2) and (3) mirrors that of SA and can be introduced by considering a continuous-time stochastic process operating on the state space $\mathcal{X}$ (or on a relaxation of $\mathcal{X}$). More precisely, following Kadowaki and Nishimori (1998) and Albash and Lidar (2018), the QA algorithm constitutes a stochastic algorithm for the minimization of a quadratic form $\boldsymbol{y}^\top J \boldsymbol{y}$, where decision variables are required to take binary values $\boldsymbol{y} \in \{0,1\}^{\tilde{n}}$ (namely, a QUBO problem). Therefore, the QA algorithm can tackle unconstrained penalty reformulations of problems (2) and (3). To construct these reformulations, we define

$$\begin{aligned} S_G &= \lfloor \log_2 n^2 \alpha_{\max} + n\beta_{\max} \rfloor + 1, \\ S_2 &= \lfloor \log_2 \tau_{\min} \rfloor + 1, \\ S_3 &= \lfloor \log_2 \tau_{\max} \rfloor + 1, \\ S_4 &= \lfloor \log_2 \theta_{\max} \rfloor + 1, \end{aligned}$$

and introduce the binary variables $s^{(G)}_k$ (for $k = 1, \ldots, S_G$), $s^{(0)}_{k,c}$ (for $k = 1, \ldots, S_G$, $c \in \mathcal{M}$), $s^{(2)}_k$ (for $k = 1, \ldots, S_2$, $c \in \mathcal{M}$), $s^{(3)}_k$ (for $k = 1, \ldots, S_3$, $c \in \mathcal{M}$), and $s^{(4)}_k$ (for $k = 1, \ldots, S_4$, $c,\ c' \in \mathcal{M}$). We denote with $\boldsymbol{s}$ the vector associated with this collection of variables. The unconstrained penalty reformulation of (2) and (3) results in a QUBO problem with objective function

$$\mathcal{H}^\mu_*(\boldsymbol{y}) = \boldsymbol{y}^\top J^\mu \boldsymbol{y} = \begin{cases} -G_1(\boldsymbol{y}) - z_1 W_1(\boldsymbol{y}) - z_2 W_2(x_{i,c}) - z_3 W_3(\boldsymbol{y}) - \lambda_4 W_4(\boldsymbol{y}) & \text{if } \mu = 1 \\ G_\infty(\boldsymbol{y}) - z_0 W_0(\boldsymbol{y}) - z_1 W_1(\boldsymbol{y}) - z_2 W_2(x_{i,c}) - z_3 W_3(\boldsymbol{y}) - \lambda_4 W_4(\boldsymbol{y}) & \text{if } \mu = \infty, \end{cases}$$

where $\boldsymbol{y}$ is the vector containing the nm original binary variables $\boldsymbol{x}$ and the S new variables $\boldsymbol{s}$ required to represent inequality constraints (6b), (7a), (7b), and (7c) as equality constraints involving binary variables only; $W_0(\boldsymbol{y})$, $W_1(\boldsymbol{y})$, $W_2(\boldsymbol{y})$, $W_3(\boldsymbol{y})$ and $W_4(\boldsymbol{y})$ correspond to the penalty terms

for constraints (6b) (7a), (7b), and (7c), with penalty weights z_0, z_1, z_2, z_3, and z_4, respectively; J^μ is a $\tilde{n}$-dimensional square matrix, whose coefficients are constructed from $G_\mu(\boldsymbol{y})$ and from $W_0(\boldsymbol{y})$, $W_1(\boldsymbol{y})$, $W_2(\boldsymbol{y})$, $W_3(\boldsymbol{y})$, and $W_4(\boldsymbol{y})$, based on the following definitions:

$$G_\mu := \begin{cases} \sum\limits_{c\in\mathcal{M}}\sum\limits_{i\in\mathcal{N}}\sum\limits_{j\in\mathcal{N}} \alpha_{i,j}x_{i,c}x_{j,c} + \sum\limits_{i\in\mathcal{N}} \beta_{i,c}x_{i,c} & \text{if } \mu = 1, \\ \sum\limits_{k=0}^{S_G} 2^k s_k^{(G)} & \text{if } \mu = \infty, \end{cases}$$

for the objectives (5a) and (6a), and

$$W_0 := \left(G_\infty(\boldsymbol{y}) - \Big(\sum_{k=0}^{S_0} 2^k s_{k,c}^{(0)} + \sum_{i\in\mathcal{N}} \Big(\sum_{j\in\mathcal{N}} \alpha_{i,j}x_{i,c}x_{j,c} + \beta_{i,c}x_{i,c} \Big) \Big) \right)^2, \quad \text{for the penalty of (6b)},$$

$$W_1 := \sum_{i\in\mathcal{N}} \left(1 - \sum_{c\in\mathcal{M}} x_{i,c} + 2 \sum_{c\in\mathcal{M}} \sum_{c'>c\in\mathcal{M}} x_{i,c}x_{i,c'} \right)^2, \quad \text{for the penalty of (7a)},$$

$$W_2 := \sum_{c\in\mathcal{M}} \left(\tau^{\min} - \Big(\sum_{i\in\mathcal{N}} x_{i,c} - \sum_{k=0}^{S_2} 2^k s_{k,c}^{(2)} \Big) \right)^2, \quad \text{for the penalty of (7b)},$$

$$W_3 := \sum_{c\in\mathcal{M}} \left(\tau^{\max} - \Big(\sum_{i\in\mathcal{N}} x_{i,c} + \sum_{k=0}^{S_3} 2^k s_{k,c}^{(3)} \Big) \right)^2, \quad \text{for the penalty of (7c)},$$

$$W_4 := \sum_{f\in\mathcal{F}} \sum_{c\in\mathcal{M}} \sum_{c'\in\mathcal{M}:c'\neq c} \left(\theta_f - \Big(\sum_{i\in\mathcal{N}} \gamma_i^f (x_{i,c} - x_{i,c'}) + \sum_{k=0}^{S_4} 2^k s_{k,c,c',f}^{(4)} \Big) \right)^2, \quad \text{for the penalty of (7d)}.$$

It is important to note that the number of binary ancillary variables required to construct this unconstrained penalty reformulation of R-MN-SAP is $S = m(S_2 + S_3) + m(m-1)S_4$, for (2), and $S = 2S_G + m(S_2 + S_3) + m(m-1)S_4$, for (3).[4]

Building on this penalty reformulation, the goal of the QA procedure is to construct a probability distribution over binary vectors in $\{0,1\}^{|\tilde{n}|}$,[5] where $\tilde{n} = nm + S$, and define the continuous-time stochastic process $\{|\Psi(t)|^2 : t \geq 0\}$, whose dynamics is governed by the differential equation

$$\tilde{i}\hbar \frac{\partial \Psi(t)}{\partial t} = H(t, \boldsymbol{y})\Psi(t), \tag{17}$$

where $\tilde{i}$ is the imaginary constant, $\hbar$ is an exogenous quantity (the reduced Planck constant), and $H : \mathbb{R} \times \{0,1\}^{2^{|\tilde{n}|}} \to \mathbb{R}$ is referred to as the Hamiltonian function,[6] satisfying

$$\begin{aligned} H(0, \boldsymbol{y}) &= \sum_{i=1}^{\tilde{n}} \mathbb{I}_\otimes^{i-1} \otimes X(\tilde{x}_i) \otimes \mathbb{I}_\otimes^{\tilde{n}-i}, \\ \lim_{t\to\infty} H(t, \boldsymbol{y}) &= \sum_{i=1}^{\tilde{n}-1} \sum_{j=i+1}^{\tilde{n}} (J_{i,j}^\mu + J_{j,i}^\mu) \mathbb{I}_\otimes^{i-1} \otimes Z(\tilde{x}_i) \otimes \mathbb{I}_\otimes^{j-i-1} Z(\tilde{x}_j) \otimes \mathbb{I}_\otimes^{\tilde{n}-j}, \end{aligned}$$

[4] In the case of problem (3) the inclusion of further ancillary variables is required to replace the products of four variables in $W_0(\boldsymbol{y})$ by the products of two variables only.

[5] To emphasize the relaxation, it is worth noting that for each $\boldsymbol{x} \in \mathcal{X}$, there exist $\boldsymbol{s}$ such that $(\boldsymbol{x}, \boldsymbol{s}) \in \{0,1\}^{|\tilde{n}|}$

[6] The dynamics this time-dependent Hamiltonian function is theoretically governed by the universal model of adiabatic evolution. This establishes that if a quantum system is subjected to a slow and continuous change in its Hamiltonian (i.e., from $H(t, \boldsymbol{y})$ to $H(t + \Delta, \boldsymbol{y})$, for some $\Delta \approx 0$), then the system will remain in its instantaneous state during the change (Kadowaki and Nishimori, 1998; Albash and Lidar, 2018).

where $\mathbb{I}_{\otimes}^{i} = I \otimes I \otimes \ldots \otimes I \in \{0,1\}^{2^i \times 2^i}$ is the tensor product of i two-dimensional identity matrices, I is a two-dimensional identity matrix, and $Z(\tilde{x}_i)$ and $X(\tilde{x}_i)$ are two-dimensional identity matrices (known as Pauli matrices) defined as

$$Z(\tilde{x}_i) = \begin{cases} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} & \text{if } \tilde{x}_i = 0 \\ \begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix} & \text{if } \tilde{x}_i = 1, \end{cases} \quad \text{and} \quad X(\tilde{x}_i) = \begin{cases} \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} & \text{if } \tilde{x}_i = 0 \\ \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} & \text{if } \tilde{x}_i = 1, \end{cases}$$

The advantage of approaching the solution of problems (2) and (3) by the continuous-time stochastic process $\{|\Psi(t)|^2 : t \geq 0\}$ lies in the unique capabilities of quantum superposition. In fact, unlike classical simulation methods, the QA algorithm allows a system of quantum particles to naturally evolve over time in accordance with (17), without making measurements (i.e., simulating from it). This unmeasured evolution allows generating the whole process $\{|\Psi(t)|^2 : t \geq 0\}$, without the computational need to run multiple simulations to attain an empirical approximation of its limit distribution (as it would be the case of SA).

In the QA hardware, each decision variable of a QUBO is mapped to a connected graph (called a Pegasus graph). The edges of this graph represent the interactions between qubits, and the vertices represent the regions on the chip where qubits are placed, so that pairs of variables in a quadratic term are associated with corresponding pairs of adjacent qubits (Serra et al., 2022). Qubits themselves are physically implemented using superconducting materials (which exhibit unique quantum behaviors at extremely low temperatures) and located at the physical vertices of the Pegasus graph. This hardware structure has already been implemented experimentally by the D-wave quantum annealers,[7] and can be seen as an analog computer for the *unmeasured generation* of the continuous-time stochastic process $\{|\Psi(t)|^2 : t \geq 0\}$ (namely, a physical representation of the dynamic of $\{|\Psi(t)|^2 : t \geq 0\}$, without the need of simulating a large number of realizations).

6 Illustrative example

We provide a simplified illustrative example of problems (2) and (3) for a case of $n = 3$ students and $m = 2$ groups. We consider the following matrix parametrization:

$$A = \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix},$$

with $\tau_{\min} = 1$ and $\tau_{\max} = 2$. For the sake of simplicity, we drop the homogeneity constraints. The resulting problems involve 6 binary variables: $x_{1,1}$, $x_{2,1}$, $x_{3,1}$, $x_{1,2}$, $x_{2,2}$, $x_{3,2}$. Using the equality constraints (7a), we remove variables $x_{1,1}$, $x_{2,2}$, and $x_{3,2}$, resulting in the following reformulations:

$$\begin{cases} \max & \eta_1 + \eta_2 \\ \text{subj. to} & \eta_1 \leq 1 - 3x_{2,1}x_{3,1} \\ & \eta_2 \leq x_{3,1} + x_{2,1} - 2x_{2,1}x_{3,1} \\ & x_{2,1},\ x_{3,1} \ \text{ binary}, \end{cases} \tag{18}$$

[7] This is Canadian quantum computing company that offers the use of commercially available quantum annealing devices. These devices are based on the physical implementation of superconducting flux qubits with Josephson Junctions Josephson (1962); Kwon et al. (2021); Krantz et al. (2019).

for problem (2), and

$$\begin{cases} \max & \eta \\ \text{subj. to} & \eta \leq 1 - 3x_{2,1}x_{3,1} \\ & \eta \leq x_{3,1} + x_{2,1} - 2x_{2,1}x_{3,1} \\ & x_{2,1},\ x_{3,1} \ \text{ binary}, \end{cases} \tag{19}$$

for problem (3). Building on the linearization (8a)-(8e), we include variable $w^1_{2,3}$ to linearize the product $x_{2,1}x_{3,1}$. The resulting systems of linear constraints for problems (18) and (19) are:

$$\begin{cases} \eta_1 \leq 1 - 3w^1_{2,3} \\ \eta_2 \leq x_{3,1} + x_{2,1} - 2w^1_{2,3} \\ w^1_{2,3} \geq x_{3,1} + x_{2,1} - 1 \\ w^1_{2,3} \leq x_{2,1} \\ w^1_{2,3} \leq x_{3,1} \end{cases} \tag{20}$$

and

$$\begin{cases} \eta \leq 1 - 3w^1_{2,3} \\ \eta \leq x_{3,1} + x_{2,1} - 2w^1_{2,3} \\ w^1_{2,3} \geq x_{3,1} + x_{2,1} - 1 \\ w^1_{2,3} \leq x_{2,1} \\ w^1_{2,3} \leq x_{3,1} \end{cases} \tag{21}$$

respectively. The detailed algebraic steps for the construction of these reformulations are provided in Appendix B. These are quadratically constrained problems.

Problem (18) can be reformulated as a QUBO problem with no addition of binary slack variables:

$$\min \quad (3x_{2,1}x_{3,1} - 1) - (x_{3,1} + x_{2,1} - 2x_{2,1}x_{3,1}), \tag{22}$$

so that

$$\boldsymbol{y}^\top J^1 \boldsymbol{y} = \begin{pmatrix} x_{2,1} \\ x_{3,1} \end{pmatrix}^\top \begin{pmatrix} -1 & 2.5 \\ 2.5 & -1 \end{pmatrix} \begin{pmatrix} x_{2,1} \\ x_{3,1} \end{pmatrix}. \tag{23}$$

Conversely, focusing on the QUBO reformulation of problem (19), the constraints involving η give rise to terms of order higher than two. Hence, the QUBO reformulation of problem (19) requires the intermediary stage of problem (21). The latter can be then transformed into a QUBO problem through the addition of seven binary slack variables s_1, s_2, ..., s_7, and two additional binary variables $s^{(G)}_1$ and $s^{(G)}_1$ to encode η. For the sake of simplicity, we make the penalty terms z_i equal to $\lceil \|A\| \rceil = 3$. We have

$$\begin{cases} \min & -s^{(G)}_1 - 2s^{(G)}_2 \\ & +3(s_5 + w^1_{2,3} - x_{2,1})^2 + 3(s_6 + w^1_{2,3} - x_{3,1})^2 + 3(w^1_{2,3} - s_7 - x_{2,1} - x_{3,1} + 1)^2 \\ & +3(s^{(G)}_1 + 2s^{(G)}_2 + s_1 + 2s_2 + 3w^1_{2,3} - 1)^2 \\ & +3(s^{(G)}_1 + 2s^{(G)}_2 + s_3 + 2s_4 - x_{3,1} - x_{2,1} + 2w^1_{2,3})^2. \end{cases} \tag{24}$$

From the three constraints required to linearize $x_{2,1}x_{3,1}$ by $w^1_{2,3}$ in (21), we note that only one can

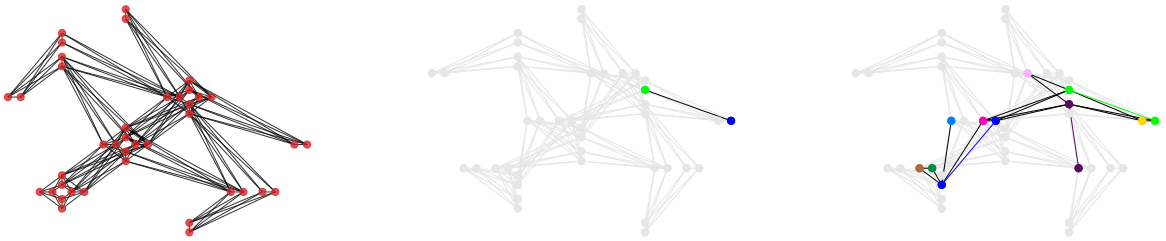
be active. Therefore, $s_5 + s_6 + s_7 = 1$, so we can eliminate one variable, obtaining

$$\begin{cases} \min & -s_1^{(G)} - 2s_2^{(G)} \\ & +3(s_5 + w_{2,3}^1 - x_{21})^2 + 3(s_6 + w_{2,3}^1 - x_{3,1})^2 + 3(s_5 + s_6 + w_{2,3}^1 - x_{2,1} - x_{3,1})^2 \\ & +3(s_1^{(G)} + 2s_2^{(G)} + s_1 + 2s_2 + 3w_{2,3}^1 - 1)^2 \\ & +3(s_1^{(G)} + 2s_2^{(G)} + s_3 + 2s_4 - x_{3,1} - x_{2,1} + 2w_{2,3}^1)^2. \end{cases} \tag{25}$$

Hence,

$$\boldsymbol{y}^\top J^\infty \boldsymbol{y} = \begin{pmatrix} s_5 \\ s_1^{(G)} \\ s_4 \\ s_2 \\ s_2^{(G)} \\ s_3 \\ s_1 \\ x_{2,1} \\ w_{2,3}^1 \\ s_6 \\ x_{3,1} \end{pmatrix}^\top \begin{pmatrix} 6 & 0 & 0 & 0 & 0 & 0 & 0 & -12 & 12 & 6 & -6 \\ 0 & 1 & 12 & 12 & 24 & 6 & 6 & -6 & 30 & 0 & -6 \\ 0 & 12 & 12 & 0 & 24 & 12 & 0 & -12 & 24 & 0 & -12 \\ 0 & 12 & 0 & 0 & 24 & 0 & 12 & 0 & 36 & 0 & 0 \\ 0 & 24 & 24 & 24 & 14 & 12 & 12 & -12 & 60 & 0 & -12 \\ 0 & 6 & 12 & 0 & 12 & 3 & 0 & -6 & 12 & 0 & -6 \\ 0 & 6 & 0 & 12 & 12 & 0 & -3 & 0 & 18 & 0 & 0 \\ -12 & -6 & -12 & 0 & -12 & -6 & 0 & 9 & -24 & -6 & 12 \\ 12 & 30 & 24 & 36 & 60 & 12 & 18 & -24 & 30 & 12 & -24 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & -6 & 12 & 6 & -12 \\ -6 & -6 & -12 & 0 & -12 & -6 & 0 & 12 & -24 & -12 & 9 \end{pmatrix} \begin{pmatrix} s_5 \\ s_1^{(G)} \\ s_4 \\ s_2 \\ s_2^{(G)} \\ s_3 \\ s_1 \\ x_{2,1} \\ w_{2,3}^1 \\ s_6 \\ x_{3,1} \end{pmatrix}. \tag{26}$$

Figure 1 provides a visual illustration of the QA hardware for problems (23) and (26), where pairs of variables in a quadratic term are associated with corresponding pairs of adjacent qubits.



(a) Overall Pegasus graph topology (b) Pegasus graph for problem (23). (c) Pegasus graph for problem (26)

Figure 1: Pegasus graph for problems (23) and (26), where edges represent interactions between qubits, and vertices represent regions on the chip where qubits are placed. These plots are obtained from the D-Wave graphical library.

Visually, the QUBO representations in problems (23) and (26) elucidate the structural distinctions between the min-sum and min-max problems under the QA procedure. In the case of the min-sum problem, the QUBO representation requires $\tilde{n} = 2$ binary variables. Conversely, for the min-max problem, the fully reduced form in (25) involves $\tilde{n} = 11$ binary variables. As numerically confirmed in Section 7, this substantial difference carries significant computational consequences. In fact, while the QA procedure emerges as the most efficient method for handling large instances of the min-sum problem (2), it also falls short of achieving optimal solutions for the min-max problem (3).

As discussed in the previous sections, our quantum and binary algorithmic integration consists in addressing the QUBO reformulation of the R-MN-SAP by the QA algorithm and the MILP reformulation by a specialized branch-and-cut algorithm, that benefits from QA solution to generate optimality bounds and Benders cuts. To illustrate the construction of a Benders cut by the QA solution of the corresponding QUBO problem, let us consider an arbitrary solution of problems (22)

and (25) (which can potentially result from a random realization of the QA algorithm):

$$x_{1,1} = 0, \quad x_{2,1} = 1, \quad x_{3,1} = 0, \quad x_{1,2} = 1, \quad x_{2,2} = 0, \quad x_{3,3} = 1, \tag{27}$$

with a min-sum value of 2 and a min-max value of 1. Note that by applying the QA procedure, no signature of optimality can be attached to this solution. However, following Mevert (1977) and Magnanti and Wong (1981), the initial selection of cuts can have a substantial impact on the performance of Benders algorithm, suggesting that an effective heuristic solution can be used not only to append lower bound to the problem, but also to establish a good selection of initial cuts.

By Proposition 3, the dual variables associated with (27) are obtained as:

$$\begin{cases} \min & x_{2,1}\overline{\zeta}_{2,3} + x_{3,1}\underline{\zeta}_{2,3} + (1 - x_{2,1} - x_{3,1})\zeta_{2,3} & \text{(28a)} \\ \text{subj. to} & \overline{\zeta}_{2,3} + \underline{\zeta}_{2,3} - \zeta_{2,3} \geq a & \text{(28b)} \\ & \overline{\zeta}_{i,j}, \underline{\zeta}_{i,j}, \zeta_{i,j} \geq 0 & \text{(28c)} \end{cases}$$

where $a = -3$, for the term $-3x_{2,1}x_{3,1}$, and $a = -2$, for the term $-2x_{2,1}x_{3,1}$ in problems (18) and (19), respectively. Replacing the solution (27), we obtain

$$\begin{cases} \overline{\zeta}_{2,3} = 0, \ \underline{\zeta}_{2,3} = 0, \ \zeta_{i,j} = 3 & \text{for the term } -3x_{2,1}x_{3,1}, \\ \overline{\zeta}_{2,3} = 0, \ \underline{\zeta}_{2,3} = 0, \ \zeta_{i,j} = 2 & \text{for the term } -2x_{2,1}x_{3,1}. \end{cases}$$

By these dual solution, problems (18) and (19) become

$$\begin{cases} \max & \eta_1 + \eta_2 \\ \text{subj. to} & \eta_1 \leq 1 + 3(1 - x_{2,1} - x_{3,1}) \\ & \eta_2 \leq x_{3,1} + x_{2,1} + 2(1 - x_{2,1} - x_{3,1}) \\ & x_{2,1}, \ x_{3,1} \ \text{ binary}, \end{cases} \tag{29}$$

and

$$\begin{cases} \max & \eta \\ \text{subj. to} & \eta \geq 1 + 3(1 - x_{2,1} - x_{3,1}) \\ & \eta \geq x_{3,1} + x_{2,1} + 2(1 - x_{2,1} - x_{3,1}) \\ & x_{2,1}, \ x_{3,1} \ \text{ binary}. \end{cases} \tag{30}$$

Table B.1 in Appendix B reports values of η_1, η_2, and η derived from problems (29) and (30), juxtaposed with the outcomes from problems (18) and (19). This comparison is exhaustive, encompassing all the possible combinations of $x_{2,1}$ and $x_{3,1}$.

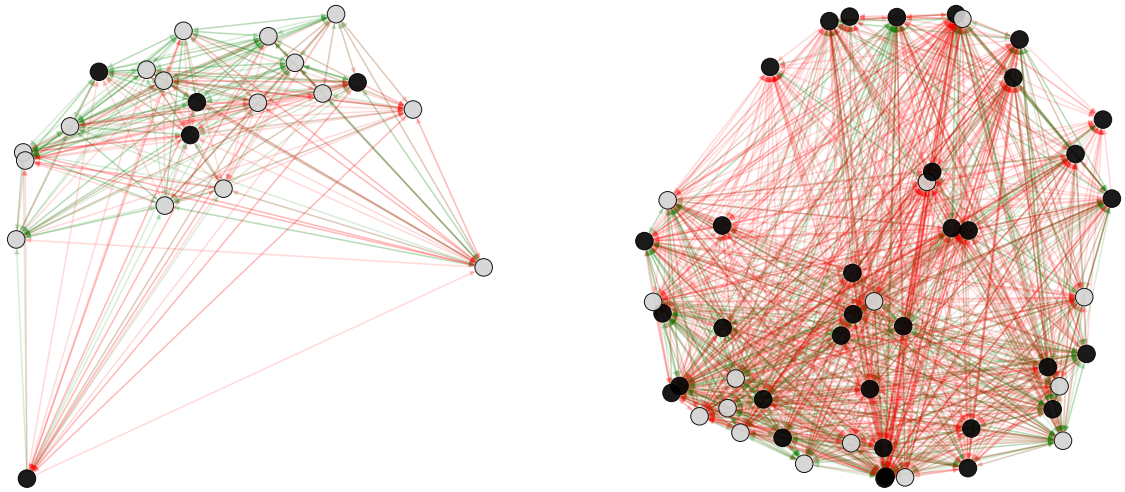
7 Empirical and computational analysis

This section provides numerical support for the theoretical framework developed thus far. This is done by constructing a large scale computational experiment involving real-world and synthetic data. The real-world data are described in Subsection 7.1, while the corresponding numerical tests are provided in Subsection 7.2.

7.1 Real-world data

We collected classroom data for group project formation in a French business school, recording students' preferences with respect to their working partners. Specifically, two classrooms are taken into account: a 21 student classroom from a postgraduate program in Digital Strategy, and a 49 student classroom in a postgraduate program in Finance. Students have been asked to assign a score of $+1$, -1, or 0 to each of their peers, resulting in pairwise preferences $\alpha_{i,j}$. As far as the set of features $\mathcal{F}$ is concerned, our data consider the students sex as a unique binary feature.

Figure 2 illustrates both network topologies with red color associated with negative edges and green color associated to positive edges. Gray nodes represent female students, while black nodes represent male students.



(a) 21 students classroom from Digital Strategy.

(b) 49 students classroom from Finance.

Figure 2: Network plot of the pairwise preferences between classroom students. The (a) and (b) panels correspond to 21 and 49 student classrooms of the Digital Strategy and the Finance postgraduate programs, respectively.

Table 1 reports the summary statistics of this data set to highlight the structural differences between the two classrooms (in terms of sizes, feature attribution, and network topology), to be used to construct numerical instances for the computational tests presented in the next subsection.

n	Density	# +	# -	Max +	Min +	Max -	Min -	M	F
21	0.69	167	123	Node 1 (17)	Node 0 (0)	Node 0 (17)	Node 8 (0)	6	15
49	0.58	499	859	Node 9 (31)	Node 47 (0)	Node 7 (34)	Node 9 (2)	34	15

Table 1: Summary statistics. From left to right, columns contain: (1) the number of students, (2) the network density, (3)-(4) the number of positive and negative links, (5) the node with the maximum number of positive links, (6) the node with the minimum number of positive links, (7) the node with the maximum number of negative links, (8) the node with the minimum number of negative links, (9)-(10) the number of male and female students.

7.2 Numerical tests

This subsection contains the results of a computational experiment involving both real-world data (presented in Subsection 7.1) and synthetic data, where problems (2) and (3) are solved using multiple algorithmic approaches. Specifically, the following solution procedures are taken into account:

- the IBM ILOG CPLEX 12.9 implementation of the branch-and-cut algorithm, based on the MILP reformulation (8b)–(8e).

- the specialized branch-and-cut algorithm (13)-(14), that appends Magnanti-Wong cuts at the integer nodes of the Branch-and-bound tree using the CPLEX Callable library.

- The D-Wave implementation of the QA algorithm applied to the QUBO reformulations of problems (2) and (3).

- A HC-based algorithm that aggregates students into groups sequentially, and restore the feasibility of the constructed solution by a sequence of pairwise exchanges.

Instances generation and comparison of solution methods. Six instances are taken into account: (i) two average size real-world instances with, $n = 21$ and $n = 49$ presented in Subsection 7.1, where students have to carry out group projects in teams of 4 to 5 members each; (ii) simulated instances with $n = 100$, $n = 200$, $n = 300$, and $n = 400$, with $m = n/10$, $\tau_{\min} = \lfloor n/m \rfloor - 1$, $\tau_{\max} = \lceil n/m \rceil + 1$, and $\theta_f = 0.7\tau_{\min}$.[8] Table C.1 (Appendix C.1) reports a detailed account of the problem sizes in terms of qubits of the QUBO reformulations and decision variables and constraints for the MILP reformulations.

For each instance, problems (2) and (3) are solved by integrating the Magnanti-Wong cuts generation and heuristic bounds within a branch-and-cut algorithm. Specifically, eight solution strategies are explored:

(0.QA) QA algorithm;

(0.HC) HC-based algorithm;

(1.a) plain branch-and-cut algorithm;

(2.a) branch-and-cut algorithm with a unique Benders cut constructed using a heuristic solution and appended at the root node;

(3.a) branch-and-cut algorithm with lower and upper bounds constraints constructed using a heuristic solution;

(1.b) Benders decomposition algorithm using Magnanti-Wong cuts;

(2.b) Benders decomposition algorithm using Magnanti-Wong cuts, with an additional Benders cut constructed using a heuristic solution and appended at the root node;

(3.b) Benders decomposition algorithm using Magnanti-Wong cuts, with a lower and upper bounds constraints constructed using a heuristic solution.

The solution strategies (2.a), (3.a), (2.b) and (3.b) integrate the heuristic solution to improve the branch-and-cut and Benders decomposition approach. When the QA method studied in Section 5 is used for this purpose, our advocate algorithmic integration between quantum and binary approaches is established. For comparative purposes, a HC-based algorithm is adopted to show that the specificity of the QA approach is to attain almost optimal solutions in few seconds, which have however, no optimality signature. Using the QA solution to generate core points, lower bounds, and initial Benders cut, allows the branch-and-cut to prune a large number of fractional and integer nodes, which can in some cases improve the QA solution.

[8] It is worth noting that our largest simulated instances (namely $n = 400$ and $m = 40$, corresponding to 16.000 binary variables) are larger than the ones used in the recent QSAP literature. For instance, the largest instance of Silva et al. (2021) considers $n = 150$ and $m = 50$ (corresponding to 7.500 binary variables).

HC-based algorithm for heuristic comparison. We provide hereafter a description of a HC-based algorithm which exploits the structural similarity property in Proposition 1 to assign m pairs of students to m initially empty groups. The second stage of the algorithm attaches the remaining $n - m$ students to groups in equal proportions, following a hierarchical clustering design. The obtained partitioning of the n students into m groups might not satisfy the group size constraints and the homogeneity constraints, so that a third stage of this heuristic procedure is included by performing pairwise exchanges that are meant to recover feasibility. The following is a stylized presentation of this HC-based algorithm:

0. Initialization: A list of empty sets $\mathcal{H}_c = \varnothing$, for $c \in \mathcal{M}$, $C_0 = \text{cor}(A)$, $C_1 = \text{cor}(A^\top)$.
1. For each $c \in \mathcal{M}$:
   - Let $B_c = \{i \in \mathcal{N} \ : \ \beta_{i,c} > 0\}$ and Q_c be the sub-matrix of $A + (C_0 + C_1)$, whose row and column indexes are in B_c.
   - Select the largest elements in Q_c, which is not present in $\mathcal{H}_c$ and store it in $\mathcal{H}_c$.
2. Repeat till all students have been assigned:

   For each $c \in \mathcal{M}$:

   Assigned to group c the individual resulting in the highest increase in utility and store it in $\mathcal{H}_c$.
3. Restore feasibility by pairwise exchanges.
4. Improved the obtained feasible solution by pairwise exchanges.

The notation $C = \text{cor}(A)$ refers to the correlation matrix between the columns of A, which is used to encode the structural similarities between students' preferences. This HC-based procedure combines a constructive greedy method for attaining a feasible solution of the R-MN-SAP (Steps 1-3) with a local search to improve the obtained solution (Step 4). In step 1, m pairs of students are selected as the initial seeds of the m groups. Step 2 represents the aggregative part, in which the remaining $n - m$ students are attached to the groups. The use of structural similarity to select the initial seeds boosts the increase of the objective function when students are attached to groups with structurally similar members.

Heuristic comparison. Our first analysis is the direct comparison between the QA algorithm and the HC-based algorithm, in terms of solution quality and computational time.

This is summarized in Table 2, listing the execution of the twelves instances: six instances of R-MN-SAP with the min-sum objective (problem (2)), and six instances of R-MN-SAP with the min-max objective (problem (3)). It is worth noting that in 100% of these instances, the QA algorithm outperforms the HC-based algorithm both in terms of solution quality and computational time. This is true for both problems (2) and (3). Noteworthy, when examining the computational times, a nuanced pattern emerges. For smaller instances (those comprising fewer than 200 students), the HC-based heuristics exhibits a computational advantage, consuming less time than the QA procedure. However, this dynamic undergoes a significant reversal for larger instance sizes (300 and 400 students), where the quantum approach not only surpasses the HC-based heuristics in solution quality but achieves this feat in a tiny fraction of the time. This temporal efficiency of the QA procedure for larger instances showcases its scalability properties. At the same time, the very large gaps in the solution qualities suggest that its application to the QUBO reformulations is capable of attaining almost optimal solutions.

	min-sum problem (2)		min-max problem (3)	
n	Obj-HC (Time)	Obj-QA (Time)	Obj-HC (Time)	Obj-QA (Time)
21 (real-world data)	15 (0.0)	60 (5.1 - 5.0)	3 (0.0)	14 (5.1 - 5.0)
49 (real-world data)	-11 (0.3)	102 (5.1 - 5.0)	-40 (0.3)	1 (5.1 - 5.0)
100	58 (1.8)	428 (5.3 - 5.0)	-3 (1.8)	7 (5.1 - 5.0)
200	203 (15.7)	823 (5.1 - 5.0)	-5 (16.6)	3 (5.1 - 5.0)
300	370 (68.9)	1422 (15.4 - 12.8)	0 (71.7)	3 (15.3 - 12.8)
400	572 (314.5)	1555 (30.5 - 12.7)	0 (315.4)	2 (30.3 - 12.7)

Table 2: Comparison between heuristic methods. From left to right, the columns contain: (i) the instance size (number of students), (ii) the HC-based algorithm applied to the min-sum problem (2), (iii) the QA algorithm applied to the min-sum problem (2), (iv) the HC-based algorithm applied to the min-max problem (3), (v) the QA algorithm applied to the min-max problem (3). The values in parenthesis refer to the solution time (in seconds). For the case of the QA algorithm, we report both the total solution time and the hit time required to create a QUBO representation and initialize the D-Wave server.

Tests for the quantum and binary algorithmic integration. Table 3 presents a comprehensive overview of numerical results obtained by means of diverse optimization approaches across various instance sizes. The leftmost column delineates QA's standalone outcomes, followed by results from different versions of the branch-and-bound algorithm of CPLEX, supplemented by results derived from the QA solution. Similarly, the right column of the table showcases outcomes from various Benders decomposition methods, also supplemented by results derived from the QA solution. Specifically, we report: (i) the plain approach, (ii) the inclusion of the fist Benders cut (computed on the QA solution), (iii) the inclusion of lower and upper bounds cut (computed on the QA solution), (iv) the inclusion of both fist Benders cut and lower and upper bounds cuts. These bounds are constructed as $\pm 20\%$ of the heuristic solution $+2$. Therefore, when these lower and upper bounds constraints are included, we lose the optimality signature, and the only goal is to outperform the heuristic solution. For each method, Table 3 reports the objective value and computational time (in seconds) enclosed in parentheses. In instances where no value is provided, it signifies that the method failed to attain a feasible solution within 600 seconds time limit.

The first insight from Table 3 underscores the dominance of the QA algorithm in addressing the min-sum problem, surpassing all other optimization methods. Notably, for larger instance sizes ($n \geq 100$), the different variants of the Benders decomposition method outperform the plain branch-and-bound algorithm of CPLEX. Remarkably, in the case of large instances ($n = 300$ and $n = 400$), where branch-and-bound faltered to yield a feasible solution, the Benders decomposition methods exhibited success. However, this dynamics shifts substantially when focusing on the min-max problem, where the different variants of the Benders decomposition (enriched with cuts from the QA solution) outperform the standalone QA and branch-and-bound. This distinction in performance can be attributed to the inherent difficulty disparity between the min-sum and min-max problems, as elucidated in the illustrative example in Section 6, where the QUBO reformulation of the min-max problem has a substantially larger dimensionality.

Table 4 serves as a comparative analysis, presenting analogous results as in Table 3 with the distinction that the HC-based algorithm is used in place of the QA algorithm to provide initial solutions for the construction of core-points for the Magnanti-Wong cuts, the initial Benders cut and the lower and upper bounds constraints. The comparison reveals a notable disparity in solution quality, with QA consistently providing substantially superior results for both the min-sum and min-max problems. The discrepancy is particularly pronounced in the context of the min-sum problem.

The trend observed in Table 3 persists when incorporating cuts derived from the HC-based

			B&B algorithm				Benders decomposition algorithm			
	n	QA	plain	1st-cut	LB	LB + 1st-cut	plain	1st-cut	LB	LB + 1st-cut
min-sum (2)	21	60	60 (8.7)	60 (22.3)	60 (23.1)	60 (56.0)	59 (600.0)	53 (600.0)	56 (600.0)	51 (600.0)
	49	102	50 (600.0)	24 (600.0)	– (600.0)	– (600.0)	9 (600.0)	9 (600.0)	– (600.0)	– (600.0)
	100	428	36 (600.0)	11 (600.0)	– (600.0)	– (600.0)	82 (600.0)	92 (600.0)	– (600.0)	– (600.0)
	200	823	3 (600.0)	-240280 (600.0)	– (600.0)	– (600.0)	237 (600.0)	209 (600.0)	– (600.0)	– (600.0)
	300	1422	– (600.0)	– (600.0)	– (600.0)	– (600.0)	335 (600.0)	225 (600.0)	– (600.0)	– (600.0)
	400	1555	– (600.0)	– (600.0)	– (600.0)	– (600.0)	159 (600.0)	264 (600.0)	– (600.0)	– (600.0)
min-max (3)	21	14	14 (38.6)	14 (72.1)	14 (44.7)	14 (134.2)	14 (100.1)	14 (180.9)	14 (115.4)	14 (84.4)
	49	1	-5 (600.0)	-3 (600.0)	1 (600.0)	0 (600.0)	-2 (600.0)	-2 (600.0)	– (600.0)	0 (600.0)
	100	7	-2970 (600.0)	-2970 (600.0)	– (600.0)	– (600.0)	5 (600.0)	3 (600.0)	9 (600.0)	10 (24.6)
	200	3	–2014 (600.0)	–2014 (600.0)	– (600.0)	– (600.0)	0 (600.0)	– (600.0)	6 (48.6)	6 (16.1)
	300	3	– (600.0)	– (600.0)	– (600.0)	– (600.0)	-8 (600.0)	-5 (600.0)	6 (376.5)	6 (299.1)
	400	2	– (600.0)	– (600.0)	– (600.0)	– (600.0)	-9 (600.0)	-5 (600.0)	1 (600.0)	4 (530.9)

Table 3: Solving the min-sum problem (2) and the min-max problem (3) with QA. We set 600 seconds as time limit.

			B&B algorithm				Benders decomposition algorithm			
	n	HC	plain	1st-cut	LB	LB + 1st-cut	plain	1st-cut	LB	LB + 1st-cut
min-sum (2)	21	15	60 (9.2)	60 (23.0)	20 (0.2)	20 (0.5)	55 (600.0)	52 (600.0)	20 (0.0)	20 (0.0)
	49	-11	50 (600.0)	40 (600.0)	-7 (32.7)	-7 (32.9)	15 (600.0)	2 (600.0)	-7 (0.6)	-7 (1.1)
	100	58	36 (600.0)	-29700 (600.0)	50 (600.0)	50 (600.0)	118 (600.0)	105 (600.0)	72 (1.3)	72 (1.3)
	200	203	3 (600.0)	-240280 (600.0)	– (600.0)	– (600.0)	223 (600.0)	214 (600.0)	227 (600.0)	246 (190.4)
	300	370	– (600.0)	– (600.0)	– (600.0)	– (600.0)	306 (600.0)	368 (600.0)	– (600.0)	– (600.0)
	400	572	– (600.0)	– (600.0)	– (600.0)	– (600.0)	120 (600.0)	147 (600.0)	– (600.0)	– (600.0)
min-max (3)	21	3	14 (34.9)	14 (98.7)	6 (0.9)	6 (91.6)	13 (299.2)	13 (318.0)	6 (0.1)	6 (0.1)
	49	-40	-5 (600.0)	-9 (600.0)	-30 (11.5)	-30 (600.0)	-2 (600.0)	-2 (600.0)	-30 (0.1)	-30 (0.1)
	100	-3	-2970 (600.0)	-2970 (600.0)	-6 (600.0)	-6 (600.0)	4 (600.0)	2 (600.0)	0 (0.6)	0 (0.6)
	200	-5	-12014 (600.0)	-12014 (600.0)	– (600.0)	– (600.0)	7 (600.0)	-2 (600.0)	-2 (8.8)	-2 (7.3)
	300	0	– (600.0)	– (600.0)	– (600.0)	– (600.0)	-7 (600.0)	– (600.0)	2 (108.6)	2 (104.8)
	400	0	– (600.0)	– (600.0)	– (600.0)	– (600.0)	-9 (600.0)	-10 (600.0)	2 (387.8)	2 (303.6)

Table 4: Solving the min-sum problem (2) and the min-max problem (3) with HC. We set 600 seconds as time limit.

	n	Best strategy	Obj (Time)
min-sum (2)	21	QA	60 (5.1)
	49	QA	102 (5.1)
	100	QA	428 (5.3)
	200	QA	823 (5.1)
	300	QA	1422 (15.4)
	400	QA	1555 (30.5)
min-max (3)	21	QA	14 (5.1)
	49	QA	1 (5.1)
	100	Benders + LB(QA) + 1st-cut(QA)	10 (24.6)
	200	Benders + LB(QA) + 1st-cut(QA)	6 (16.1)
	300	Benders + LB(QA) + 1st-cut(QA)	6 (299.1)
	400	Benders + LB(QA) + 1st-cut(QA)	4 (530.9)

Table 5: Solving the min-sum problem (2) and the min-max problem (3) with HC. We set 600 seconds as time limit.

heuristic into the Benders decomposition. This enhances the solution quality, compared to the baseline ILOG-CPLEX implementation of the branch-and-cut algorithm. However, when juxtaposed with outcomes derived from Benders decomposition fueled by the QA solution, the corresponding results from Table 4 exhibit suboptimal performance across all instances and for both problem types, min-sum and min-max. These findings underscore the critical role of high-quality cuts in Benders decomposition methods, revealing the pivotal importance of leveraging the superior quality of the QA solution when generating core-points, initial cuts, and lower and upper bounds constraints.

Table 5 summarizes the results from Tables 3 and 4, highlighting the best solution strategy associated with each instance of the R-MN-SAP. The QA algorithm is by itself the best solution strategy for the min-sum problem (2). However, for large instances of the min-max problem (3), the quantum and binary algorithmic integration (labeled as "Benders + LB(QA) + 1st-cut(QA)" in Table 5) exhibits its outstanding performance among the other analyzed approaches.

8 Conclusion

In this paper, we introduced and addressed the R-MN-SAP, a specialized variant of the QSAP, with a focus on collaborative team formation in university classrooms. The problem arises from the need to efficiently organize students into groups, considering factors such as interpersonal affinity, individual preferences for group projects, and adherence to institutional standards of homogeneity and group size. Our formulation extends the traditional QSAP by incorporating a generalized objective function that allows for flexibility in utility maximization within groups, guided by a parameter μ that ranges from the min-sum to the min-max objectives. Additionally, we introduced between-group homogeneity constraints and group size constraints to address real-world scenarios, which reflect institutional requirements of university classrooms and align our modeling design to the group formation literature.

To tackle the R-MN-SAP, we proposed a hybrid algorithmic approach that integrates a specialized branch-and-cut procedure with Magnanti-Wong Benders cuts and strong optimality bounds, by means of a high quality heuristic method. To this end, we exploit the QA algorithm applied to the QUBO reformulation of our model.

Our experimental results demonstrate the effectiveness of the proposed approach in efficiently

solving large instances of the R-MN-SAP. The algorithm exhibits superior performance in terms of solution quality and computational speed compared to existing methods. As a result, the synergy between classical binary optimization techniques and quantum computing methodologies appears as a promising avenue for solving complex combinatorial optimization problems, offering a potential boost in the computational efficiency.

While our model and algorithm exhibit promising results, there are avenues for further research and improvement. Future work could explore the integration of additional features or constraints that capture the evolving dynamics of student preferences and team collaboration over time. Moreover, the scalability of our approach to accommodate even larger classroom sizes and more intricate features warrants further level of quantum and binary algorithmic integration, such as exploiting a bundle of realized QA solutions to generate initial cuts.

In conclusion, our study advances the understanding of collaborative team formation by introducing and addressing the R-MN-SAP. Our successful application of the proposed algorithmic approach to this problem holds promise for transforming how we approach and solve challenging combinatorial optimization problems from diverse domains using our novel quantum and binary algorithmic integration.

References

Albash, T., and D. A. Lidar. 2018. Adiabatic quantum computation. *Reviews of Modern Physics* 90 (1): 015002.

Axelrod, R., W. Mitchell, R. E. Thomas, D. S. Bennett, and E. Bruderer. 1995. Coalition formation in standard-setting alliances. *Management Science* 41 (9): 1493–1508.

Basso, F., L. J. Basso, M. Rönnqvist, and A. Weintraub. 2021. Coalition formation in collaborative production and transportation with competing firms. *European Journal of Operational Research* 289 (2): 569–581.

Benders, J. 2005. Partitioning procedures for solving mixed-variables programming problems. *Computational Management Science* 2:3–19. English translation of the original paper appeared in Numerische Mathematik, 4:238–252, 1962.

Bergman, D. 2019. An exact algorithm for the quadratic multiknapsack problem with an application to event seating. *INFORMS Journal on Computing* 31 (3): 477–492.

Bian, Z., F. Chudak, W. Macready, A. Roy, R. Sebastiani, and S. Varotti. 2020. Solving SAT (and MaxSAT) with a quantum annealer: Foundations, encodings, and preliminary results. *Information and computation* 275:104609.

Billionnet, A., and S. Elloumi. 2001. Best reduction of the quadratic semi-assignment problem. *Discrete Applied Mathematics* 109 (3): 197–213.

Bova, F., A. Goldfarb, and R. G. Melko. 2023. Quantum economic advantage. *Management Science* 69 (2): 1116–1126.

Burkard, R., M. Dell'Amico, and S. Martello. 2012. *Assignment problems, revised reprint*. Philadelphia, PA: SIAM.

Burkard, R. E., and F. Rendl. 1984. A thermodynamically motivated simulation procedure for combinatorial optimization problems. *European Journal of Operational Research* 17 (2): 169–174.

Carrell, S. E., B. I. Sacerdote, and J. E. West. 2013. From natural variation to optimal policy? The importance of endogenous peer group formation. *Econometrica* 81 (3): 855–882.

Carugno, C., M. Ferrari Dacrema, and P. Cremonesi. 2022. Evaluating the job shop scheduling problem on a d-wave quantum annealer. *Nature Scientific Reports* 12 (1): 6539.

Castro, J., S. Nasini, and F. Saldanha-da Gama. 2017. A cutting-plane approach for large-scale capacitated multi-period facility location using a specialized interior-point method. *Mathematical Programming* 163 (1-2): 411–444.

Cool, K. O., and D. Schendel. 1987. Strategic group formation and performance: The case of the US pharmaceutical industry, 1963–1982. *Management Science* 33 (9): 1102–1124.

Demaine, E. D., D. Emanuel, A. Fiat, and N. Immorlica. 2006. Correlation clustering in general weighted graphs. *Theoretical Computer Science* 361 (2-3): 172–187.

Drwal, M. 2014. Algorithm for quadratic semi-assignment problem with partition size coefficients. *Optimization Letters* 8 (3): 1183–1190.

Fischetti, M., I. Ljubić, and M. Sinnl. 2017. Redesigning Benders decomposition for large-scale facility location. *Management Science* 63 (7): 2146–2162.

Gallo, G., and B. Simeone. 1991. Optimal grouping of researchs into departments. *Ricerca Operativa* (1991/57).

Glover, F., and E. Woolsey. 1974. Converting the 0-1 polynomial programming problem to a 0-1 linear program. *Operations Research* 22 (1): 180–182.

Glover (a), F., G. Kochenberger, R. Hennig, and Y. Du. 2022. Quantum bridge analytics I: a tutorial on formulating and using QUBO models. *Annals of Operations Research* 314 (1): 141–183.

Glover (b), F., G. Kochenberger, M. Ma, and Y. Du. 2022. Quantum Bridge Analytics II: QUBO-Plus, network optimization and combinatorial chaining for asset exchange. *Annals of Operations Research* 314 (1): 185–212.

Greenberg, H. 1969. A quadratic assignment problem without column constraints. *Naval Research Logistics Quarterly* 16 (3): 417–421.

Josephson, B. D. 1962. Possible new effects in superconductive tunnelling. *Physics letters* 1 (7): 251–253.

Jouida, S. B., S. Krichen, and W. Klibi. 2017. Coalition-formation problem for sourcing contract design in supply networks. *European Journal of Operational Research* 257 (2): 539–558.

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

Kirkpatrick, S., C. D. Gelatt Jr, and M. P. Vecchi. 1983. Optimization by simulated annealing. *science* 220 (4598): 671–680.

Koopmans, T. C., and M. Beckmann. 1957. Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*:53–76.

Krantz, P., M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver. 2019. A quantum engineer's guide to superconducting qubits. *Applied physics reviews* 6 (2): 021318.

Kwon, S., A. Tomonaga, G. Lakshmi Bhai, S. J. Devitt, and J.-S. Tsai. 2021. Gate-based superconducting quantum computing. *Journal of Applied Physics* 129 (4): 041102.

Lawler, E. L. 1963. The quadratic assignment problem. *Management Science* 9 (4): 586–599.

Magirou, V., and J. Milis. 1989. An algorithm for the multiprocessor assignment problem. *Operations Research letters* 8 (6): 351–356.

Magnanti, T., and R. Wong. 1981. Accelerating Benders decomposition: algorithmic enhancement and model selection criteria. *Operations Research* 29:464–484.

Malucelli, F. 1996. A polynomially solvable class of quadratic semi-assignment problems. *European Journal of Operational Research* 91 (3): 619–622.

Malucelli, F., and D. Pretolani. 1994. Quadratic semi-assignment problems on structured graphs. *Ricerca Operativa* 69:57–78.

Malucelli, F., and D. Pretolani. 1995. Lower bounds for the quadratic semi-assignment problem. *European Journal of Operational Research* 83 (2): 365–375.

Mevert, P. 1977. Fixed charge network flow problems: Applications and methods of solution. In *Large Scale and Hirarchical Systems Workshop, Brussels*.

Milliken, F. J., and L. L. Martins. 1996. Searching for common threads: Understanding the multiple effects of diversity in organizational groups. *Academy of management review* 21 (2): 402–433.

Nannicini, G. 2022. Fast quantum subroutines for the simplex method. *Operations Research*.

Neshati, M., H. Beigy, and D. Hiemstra. 2014. Expert group formation using facility location analysis. *Information processing & management* 50 (2): 361–383.

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

Olivier, P., A. Lodi, and G. Pesant. 2021. The quadratic multiknapsack problem with conflicts and balance constraints. *INFORMS Journal on Computing* 33 (3): 949–962.

Papadakos, N. 2008. Practical enhancements to the Magnanti–Wong method. *Operations Research Letters* 36 (4): 444–449.

Parekh, O. 2023. Synergies Between Operations Research and Quantum Information Science. *INFORMS Journal on Computing* 35 (2): 266–273.

Pollalis, Y. A., and G. Mavrommatis. 2009. Using similarity measures for collaborating groups formation: A model for distance learning environments. *European Journal of Operational Research* 193 (2): 626–636.

Pycia, M. 2012. Stability and preference alignment in matching and coalition formation. *Econometrica* 80 (1): 323–362.

Rei, W., J.-F. Cordeau, M. Gendreau, and P. Soriano. 2009. Accelerating Benders decomposition by local branching. *INFORMS Journal on Computing* 21:333–345.

Sahni, S., and T. Gonzalez. 1976. P-complete approximation problems. *Journal of the ACM (JACM)* 23 (3): 555–565.

Saito, H., T. Fujie, T. Matsui, and S. Matuura. 2009. A study of the quadratic semi-assignment polytope. *Discrete Optimization* 6 (1): 37–50.

Serra, T., T. Huang, A. U. Raghunathan, and D. Bergman. 2022. Template-based minor embedding for adiabatic quantum optimization. *INFORMS Journal on Computing* 34 (1): 427–439.

Sherali, H. D., and W. P. Adams. 1998. Reformulation-linearization techniques for discrete optimization problems. *Handbook of Combinatorial Optimization: Volume1–3*:479–532.

Silva, A., L. C. Coelho, and M. Darvish. 2021. Quadratic assignment problem variants: A survey and an effective parallel memetic iterated tabu search. *European Journal of Operational Research* 292 (3): 1066–1084.

Stollenwerk, T., B. O'Gorman, D. Venturelli, S. Mandra, O. Rodionova, H. Ng, B. Sridhar, E. G. Rieffel, and R. Biswas. 2019. Quantum annealing applied to de-conflicting optimal trajectories for air traffic management. *IEEE transactions on intelligent transportation systems* 21 (1): 285–297.

Wentges, P. 1996. Accelerating Benders' decomposition for the capacitated facility location problem. *Mathematical Methods of Operations Research* 44:267–290.

A : Mathematical proofs

Proposition 1

Proof. Without loss of generality let $\mathcal{M} = \{c, c'\}$ and $x_h = \mathbb{1}\{\lambda(h) = c\}$, where $\mathbb{1}$ is the indicator function. We consider two cases:

- if $\lambda(i) = \lambda(j) = c$ (namely, i and j are assigned to the same group), then:

$$\begin{cases} \delta_c(\lambda) & = \kappa - \alpha_{i,j} - \alpha_{j,i} - \beta_{i,c} - \beta_{j,c} - 2\ell(\boldsymbol{x}_{-\{i,j\}}) - g(\boldsymbol{x}_{-\{i,j\}}) - q_c(\boldsymbol{x}_{-\{i,j\}}), \\ \delta_{c'}(\lambda) & = \kappa - g(1 - \boldsymbol{x}_{-\{i,j\}}) - q_{c'}(1 - \boldsymbol{x}_{-\{i,j\}}), \end{cases}$$

where

$$\begin{cases} \ell(\boldsymbol{x}_{-\{i,j\}}) & = \sum_{h \in \mathcal{N}\setminus\{i,j\}} (\alpha_{i,h} + \alpha_{h,i}) x_h \\ g(\boldsymbol{x}_{-\{i,j\}}) & = \sum_{h \in \mathcal{N}\setminus\{i,j\}} \sum_{s \in \mathcal{N}\setminus\{i,j\}} \alpha_{h,s} x_h x_s, \\ q_c(\boldsymbol{x}_{-\{i,j\}}) & = \sum_{h \in \mathcal{N}\setminus\{i,j\}} \beta_{h,c} x_h. \end{cases}$$

Hence, the total disutility is

$$\begin{aligned} U(\boldsymbol{x}_{-\{i,j\}}) &= \kappa - \alpha_{i,j} - \alpha_{j,i} - \beta_{i,c} - \beta_{j,c} - 2\ell(\boldsymbol{x}_{-\{i,j\}}) - g(\boldsymbol{x}_{-\{i,j\}}) - g(1 - \boldsymbol{x}_{-\{i,j\}}) \\ &\quad - q_c(\boldsymbol{x}_{-\{i,j\}}) - q_{c'}(1 - \boldsymbol{x}_{-\{i,j\}}), \\ &= \kappa - \alpha_{i,j} - \alpha_{j,i} - \beta_{i,c} - \beta_{j,c} - 2 \sum_{h \in \mathcal{N}\setminus\{i,j\}} (\alpha_{h,i} + \alpha_{i,h}) x_h \\ &\quad - 2 \sum_{h \in \mathcal{N}\setminus\{i,j\}} \sum_{s \in \mathcal{N}\setminus\{i,j\}} \alpha_{h,s} x_h x_s + \sum_{h \in \mathcal{N}\setminus\{i,j\}} (\overline{\alpha}_h^{(i,j)} + \underline{\alpha}_h^{(i,j)}) x_h \\ &\quad - \sum_{h \in \mathcal{N}\setminus\{i,j\}} (\beta_{c,h} - \beta_{c',h}) x_h - \sum_{h \in \mathcal{N}\setminus\{i,j\}} \sum_{s \in \mathcal{N}\setminus\{i,j\}} \alpha_{h,s} + \sum_{h \in \mathcal{N}\setminus\{i,j\}} \beta_{c',h} \\ &= C - 2 \sum_{h \in \mathcal{N}\setminus\{i,j\}} \sum_{s \in \mathcal{N}\setminus\{i,j\}} \alpha_{h,s} x_h x_s - \sum_{h \in \mathcal{N}\setminus\{i,j\}} \omega_h x_h \end{aligned}$$

where $\overline{\alpha}_h^{(i,j)} = \sum_{s \in \mathcal{N}\setminus\{i,j\}} \alpha_{s,h}$, $\underline{\alpha}_h^{(i,j)} = \sum_{s \in \mathcal{N}\setminus\{i,j\}} \alpha_{h,s}$, and

$$\begin{cases} \omega_h & = \beta_{h,c} - \beta_{h,c'} + \alpha_{h,i} + \alpha_{i,h} - \overline{\alpha}_h^{(i,j)} - \underline{\alpha}_h^{(i,j)} \\ C & = \kappa - \alpha_{i,j} - \alpha_{j,i} - \beta_{i,c} - \beta_{j,c} - \sum_{h \in \mathcal{N}\setminus\{i,j\}} \sum_{s \in \mathcal{N}\setminus\{i,j\}} \alpha_{h,s} + \sum_{h \in \mathcal{N}\setminus\{i,j\}} \beta_{c',h} \end{cases}$$

- if $\lambda(i) = c$ and $\lambda(j) = c'$ (namely, i is assigned to group c and j is assigned to group c'), then:

$$\begin{cases} \delta_c(\lambda) & = \kappa - \beta_{i,c} - \Big(\sum_{h \in \mathcal{N}\setminus\{i,j\}} (\alpha_{i,h} + \alpha_{h,i}) x_h \Big) - g(\boldsymbol{x}_{-\{i,j\}}) - q_c(\boldsymbol{x}_{-\{i,j\}}), \\ \delta_{c'}(\lambda) & = \kappa - \beta_{j,c'} - \Big(\sum_{h \in \mathcal{N}\setminus\{i,j\}} (\alpha_{i,h} + \alpha_{h,i})(1 - x_h) \Big) - g(1 - \boldsymbol{x}_{-\{i,j\}}) - q_{c'}(\boldsymbol{x}_{-\{i,j\}}), \end{cases}$$

Hence, the total disutility is

$$\begin{aligned}
\tilde{U}(\boldsymbol{x}_{-\{i,j\}}) &= \kappa - \beta_{i,c} - \beta_{j,c'} - \ell(\mathbf{1}) - g(\boldsymbol{x}_{-\{i,j\}}) - g(1-\boldsymbol{x}_{-\{i,j\}}) \\
&\quad -q_c(\boldsymbol{x}_{-\{i,j\}}) - q_{c'}(1-\boldsymbol{x}_{-\{i,j\}}), \\
&= \kappa - \beta_{i,c} - \beta_{j,c'} - \sum_{h\in\mathcal{N}\setminus\{i,j\}} (\alpha_{h,i}+\alpha_{i,h}) \\
&\quad -2\sum_{h\in\mathcal{N}\setminus\{i,j\}}\sum_{s\in\mathcal{N}\setminus\{i,j\}} \alpha_{h,s}x_h x_s + \sum_{h\in\mathcal{N}\setminus\{i,j\}} (\overline{\alpha}_h^{(i,j)} + \underline{\alpha}_h^{(i,j)})x_h \\
&\quad - \sum_{h\in\mathcal{N}\setminus\{i,j\}} (\beta_{c,h}-\beta_{c',h})x_h - \sum_{h\in\mathcal{N}\setminus\{i,j\}}\sum_{s\in\mathcal{N}\setminus\{i,j\}} \alpha_{h,s} + \sum_{h\in\mathcal{N}\setminus\{i,j\}} \beta_{c',h} \\
&= \tilde{C} - 2\sum_{h\in\mathcal{N}\setminus\{i,j\}}\sum_{s\in\mathcal{N}\setminus\{i,j\}} \alpha_{h,s}x_h x_s - \sum_{h\in\mathcal{N}\setminus\{i,j\}} \tilde{\omega}_h x_h
\end{aligned}$$

where

$$\begin{cases}
\tilde{\omega}_h &= \beta_{h,c} - \beta_{h,c'} - \overline{\alpha}_h^{(i,j)} - \underline{\alpha}_h^{(i,j)} \\
\tilde{C} &= \kappa - \beta_{i,c} - \beta_{j,c'} - \sum_{h\in\mathcal{N}\setminus\{i,j\}} (\alpha_{h,i}+\alpha_{i,h}) - \sum_{h\in\mathcal{N}\setminus\{i,j\}}\sum_{s\in\mathcal{N}\setminus\{i,j\}} \alpha_{h,s} + \sum_{h\in\mathcal{N}\setminus\{i,j\}} \beta_{c',h}
\end{cases}$$

Combining U and $\tilde{U}$, we obtain the following sufficient condition

$$U(\boldsymbol{x}_{-\{i,j\}}) \;\le\; \tilde{U}(\boldsymbol{x}_{-\{i,j\}}).$$

which implies

$$\begin{aligned}
C - \sum_{h\in\mathcal{N}\setminus\{i,j\}} \omega_h x_h &\le \tilde{C} - \sum_{h\in\mathcal{N}\setminus\{i,j\}} \tilde{\omega}_h x_h \\
-\alpha_{i,j} - \alpha_{j,i} - \beta_{i,c} - \beta_{j,c} - \sum_{h\in\mathcal{N}\setminus\{i,j\}} \omega_h x_h &\le -\beta_{i,c} - \beta_{j,c'} - \sum_{h\in\mathcal{N}\setminus\{i,j\}} (\alpha_{h,i}+\alpha_{i,h}) - \sum_{h\in\mathcal{N}\setminus\{i,j\}} \tilde{\omega}_h x_h \\
-\alpha_{i,j} - \alpha_{j,i} - \beta_{j,c} - \sum_{h\in\mathcal{N}\setminus\{i,j\}} (\alpha_{h,i}+\alpha_{i,h})x_h &\le -\beta_{j,c'} - \sum_{h\in\mathcal{N}\setminus\{i,j\}} (\alpha_{h,i}+\alpha_{i,h}) \\
\alpha_{i,j} + \alpha_{j,i} &\ge \beta_{i,c} - \beta_{j,c'} - \sum_{h\in\mathcal{N}\setminus\{i,j\}} (\alpha_{h,i}+\alpha_{i,h})(1-x_h) \\
\alpha_{i,j} + \alpha_{j,i} &\ge 1 - \beta_{i,c} - \sum_{h\in\mathcal{N}\setminus\{i,j\}} (\alpha_{h,i}+\alpha_{i,h})(1-x_h)
\end{aligned}$$

□

Proposition 2

Proof. For this proof we use a similar argument as in Sahni and Gonzalez (1976) and reduce the R-MN-SAP to an m-partitioning problem. Given a multiset of $\overline{n}$ positive integers $\{r_1,\ldots,r_{\overline{n}}\}$ and a positive integer m, this consists in partitioning $\{r_1,\ldots,r_{\overline{n}}\}$ into m disjoint subsets $S_1,\ldots,S_m$, such that the sum of the integers in each subset is equal:

$$\sum_{i\in S_j} r_i = \overline{r}, \quad \forall j = 1,\ldots m.$$

To lighten the notation, we construct this proof for the case $m = 6$. However, by a careful enumeration, an analogous argument can be applied to other cases. We construct the following specification of (1), based on a star graph with $n = \overline{n} + 2m$ nodes, where $\overline{n} + 1$ is the central node:

- $\mathcal{N} = \{1, \ldots, n, n+1, \ldots, \overline{n}+2m\}$, $\mathcal{E} = \{(i, \overline{n}+1) \ : \ 1 \leq i \leq \overline{n}\} \cup \left\{ \bigcup_{h=1}^{2m-1} \{(\overline{n} + 1 + h, \overline{n} + 1)\} \right\}$,

- $\alpha_{i,j} = \begin{cases} r_i & \text{if } (i,j) \in \mathcal{E},\ 1 \leq i \leq \overline{n}, \\ 1 & \text{if } (i,j) = (\overline{n} + 1 + h, \overline{n} + 1) \in \mathcal{E},\ \overline{n} + 2 \leq i \leq \overline{n} + 2m, \end{cases}$

- $\beta_{i,c} = \begin{cases} \frac{c-1}{T} r_i & \text{if } 1 \leq i \leq \overline{n}, \\ 0 & \text{if } i \in \{\overline{n} + 1, \ldots, \overline{n} + 2m\}, \end{cases}$

- $\gamma_c^f = \begin{cases} f r_i & \text{if } 1 \leq i \leq \overline{n}, \\ f \frac{T}{3} & \text{if } i \in \{\overline{n} + 1, \ldots, \overline{n} + m\}, \\ f \frac{T}{2} & \text{if } i \in \{\overline{n} + m + 1, \ldots, \overline{n} + 2m\}, \end{cases}$

- $\theta_f = 0$, $\tau_{\min} = 2$, $\tau_{\max} = n - 2m$. $T = \sum_{i=1}^{\overline{n}} r_i$,

This parametrization is represented in Figure 3.

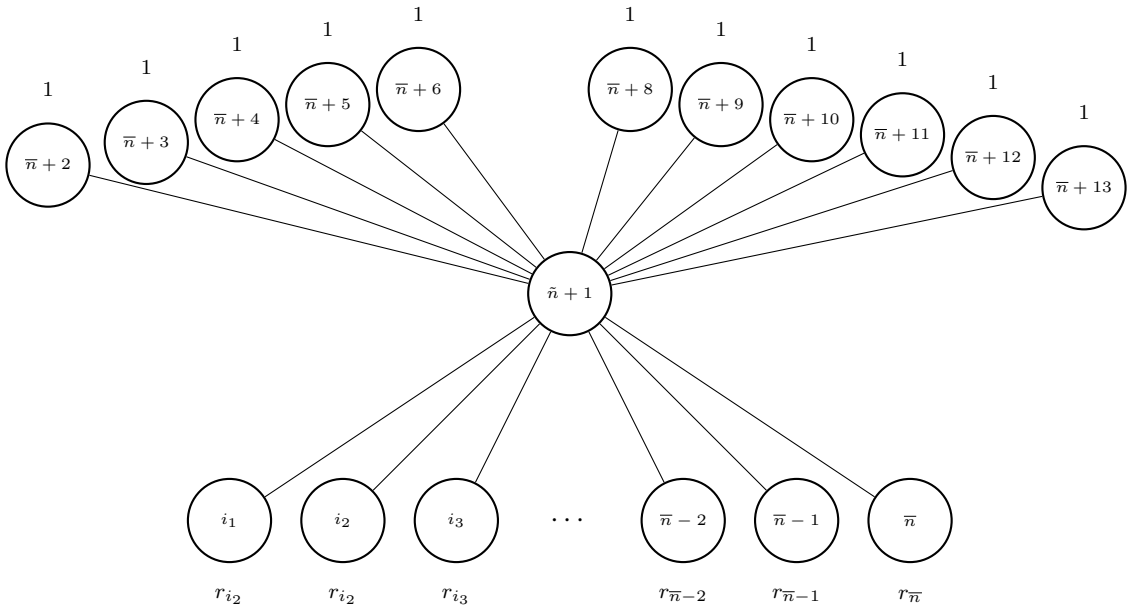


Figure 3: Star graph configuration with $m = 6$.

To prove the equivalence, we study the two implications.

Case 1. If the m-partitioning problem has a solution for $\{r_1, \ldots, r_{\overline{n}}\}$, then there exists a feasible solution of problem (1), whose objective function has value $\overline{r} B_\mu$, where

$$B_\mu = \left(1 + \sum_{h=1}^{m-1} \left(\frac{h}{T} \right)^\mu \right)^{1/\mu}.$$

To see why this is the case, let $S_1, \ldots, S_m$ be a solution of the m-partitioning problem, and construct a solution of problem (1) in which

$$\lambda(\overline{n} + h) = \lambda(\overline{n} + m + h) = h, \quad \forall h = 1, \ldots, m. \tag{A.1}$$

In other words, nodes $\overline{n}+1$ and $\overline{n}+m+1$ are assigned to group 1, nodes $\overline{n}+2$ and $\overline{n}+m+2$ are assigned to group 1, and so on and so forth. Further, all nodes from 1 up to $\overline{n}$ are assigned to group 1. Therefore, group 1 has $\overline{n}+2$ members, whereas all the other groups have 2 members. Next, for all $i \in S_h$, we set $\lambda_i = h$, for all $h = 1, \ldots, m$, in order to map the m-partitioning of $\{r_1, \ldots, r_n\}$ into a solution of problem (1). Note that each group contains at most $\overline{n} - m + 1$ members, as $S_h \neq \emptyset$, for all $h = 1, \ldots, m$. This is graphically illustrated in Figure 4.

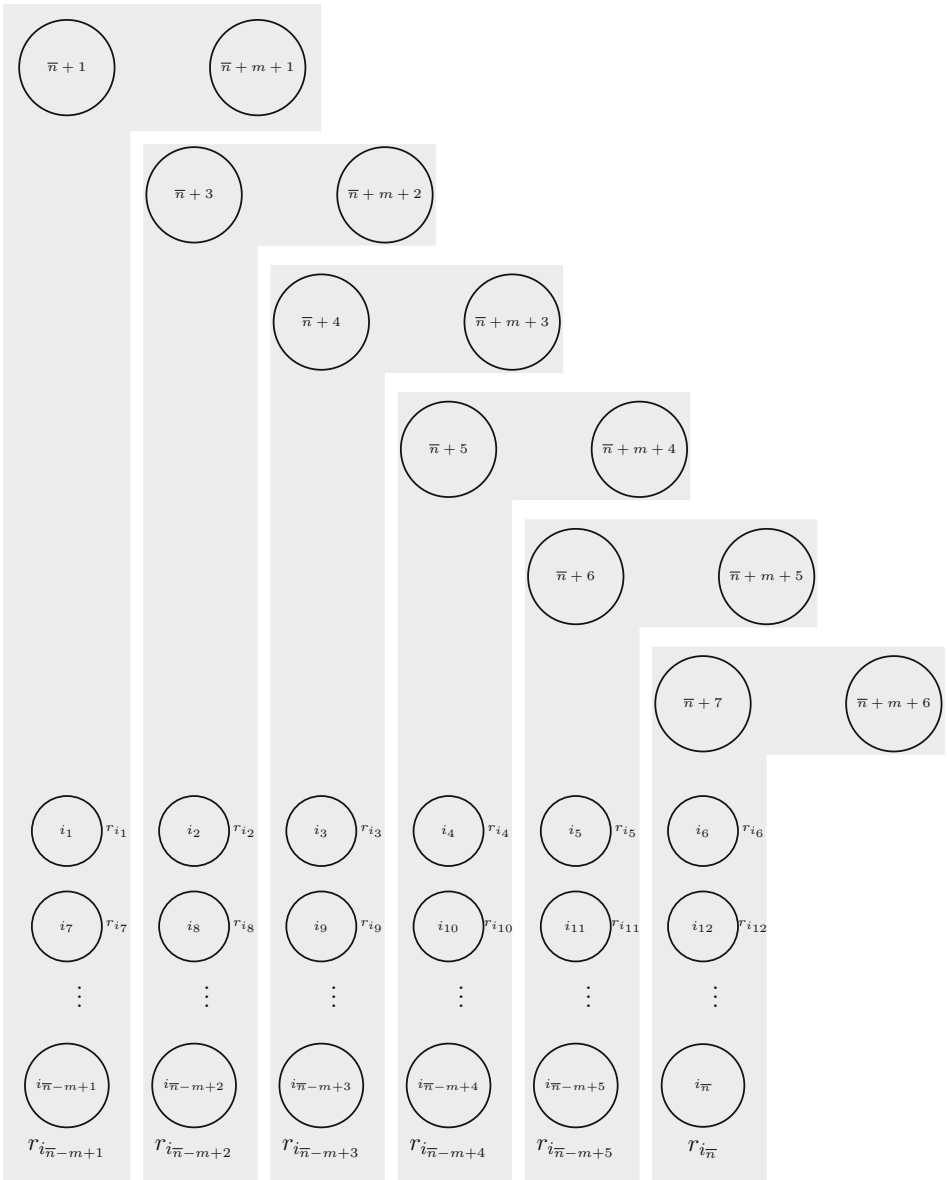


Figure 4: Solution (A.1) for the R-MN-SAP with $m = 6$, based on the parametrization in Figure 3. The six gray areas represent the groups formation.

The objective function value, associated with this partitioning becomes

$$V_{(\text{A.1})} = \underbrace{\Big(\Big(\sum_{i:\lambda(i)=1}\Big(r_i + \frac{0}{T}r_i\Big)\Big)^{\mu}}_{\text{for } c=1} + \underbrace{\Big(\sum_{i:\lambda(i)=2}\Big(0 + \frac{1}{T}r_i\Big)\Big)^{\mu}}_{\text{for } c=2} + \ldots + \underbrace{\Big(\sum_{i:\lambda(i)=m}\Big(0 + \frac{m-1}{T}r_i\Big)\Big)^{\mu}\Big)^{1/\mu}}_{\text{for } c=m}$$

$$= \left(\overline{r}^{\mu} + \left(\frac{1}{T}\right)^{\mu}\overline{r}^{\mu} + \ldots + \left(\frac{m-1}{T}\right)^{\mu}\overline{r}^{\mu}\right)^{1/\mu} = \overline{r}B_{\mu}.$$

To show that such solution is feasible, we note that $\tau_{\min} = 2 \leq |\lambda_i^{-1}(c)| \leq \overline{n} - m + 1 = \tau_{\max}$. Next, for all $c,\ c' \in \mathcal{M}$, we have

$$\left| \sum_{i \in \lambda^{-1}(c)} \gamma_i^f - \sum_{i \in \lambda^{-1}(c')} \gamma_i^f \right| = f \left| \left(\frac{T}{3} + \frac{T}{2} + \sum_{i \in \lambda^{-1}(c)} r_i \right) - \left(\frac{T}{3} + \frac{T}{2} + \sum_{i \in \lambda^{-1}(c')} r_i \right) \right|.$$

Since $\sum_{i \in \lambda^{-1}(c')} r_i = \sum_{i \in \lambda^{-1}(c)} r_i = \overline{r}$ (as we are considering the case when the m-partitioning problem has a solution), then

$$\left| \sum_{i \in \lambda^{-1}(c)} \gamma_i^f - \sum_{i \in \lambda^{-1}(c')} \gamma_i^f \right| = \left| \left(f\frac{T}{3} + f\frac{T}{2} + f\overline{r} \right) - \left(f\frac{T}{3} + f\frac{T}{2} + f\overline{r} \right) \right| \leq 0. \tag{A.2}$$

Case 2. If a solution of problem (1) has objective function greater or equal to $\overline{r}B_\mu$, this implies that the corresponding m-partitioning problem has a solution for $\{r_1, \ldots, r_n\}$, as constraint (ii) combined with $\theta_f = 0$ implies that $\sum_{i \in \lambda^{-1}(c)} \gamma_i^f = \sum_{i \in \lambda^{-1}(c')} \gamma_i^f$, for all $c,\ c' \in \mathcal{M}$. This can only be satisfied if $\{r_1, \ldots, r_n\}$ admits a m-partitioning. In fact, solution (A.1) is infeasible if $\sum_{i \in \lambda^{-1}(c')} r_i \neq \sum_{i \in \lambda^{-1}(c)} r_i = \overline{r}$, so that

$$\left| \sum_{i \in \lambda^{-1}(c)} \gamma_i^f - \sum_{i \in \lambda^{-1}(c')} \gamma_i^f \right| = f \left| \left(\frac{T}{3} + \frac{T}{2} + \sum_{i \in \lambda^{-1}(c)} r_i \right) - \left(\frac{T}{3} + \frac{T}{2} + \sum_{i \in \lambda^{-1}(c')} r_i \right) \right| > 0.$$

We proceed by constructing a family of $m!$ feasible solutions of problem (1)(different from solution (A.1)) for the case when $\sum_{i \in \lambda^{-1}(c')} r_i \neq \sum_{i \in \lambda^{-1}(c)} r_i = \overline{r}$ (namely, the m-partitioning does not exist), and show that such solutions do not attain the value $\overline{r}B_\mu$, so that the answer to μ-R-MN-SAP is negative. Each of these solutions is obtained by group permutation, so that we provide below a particular form:

$$\begin{cases} \lambda(i) = 1 & \text{for } 1 \leq i \leq \overline{n}, \\ \lambda\left(\overline{n} + (j-1)\frac{m}{3} + i\right) = j + 1 & \text{for } 1 \leq i \leq \frac{m}{3},\ 1 \leq j \leq 3 \\ \lambda\left(\overline{n} + m + (j-1)\frac{m}{2} + i\right) = 4 + j & \text{for } 1 \leq i \leq \frac{m}{2},\ 1 \leq j \leq 2 \end{cases} \tag{A.3}$$

To show that such solution is feasible, we note that $\tau_{\min} = 2 \leq |\lambda_i^{-1}(c)| \leq \overline{n} = \tau_{\max}$. Next, for all $c,\ c' \in \mathcal{M}$, we have

$$\begin{aligned} \sum_{i \in \lambda^{-1}(1)} \gamma_i^f &= f \sum_{i=1}^{\overline{n}} r_i &&= fT \\ \sum_{i \in \lambda^{-1}(2)} \gamma_i^f &= f\frac{T}{2} + f\frac{T}{2} &&= fT \\ \sum_{i \in \lambda^{-1}(3)} \gamma_i^f &= f\frac{T}{2} + f\frac{T}{2} &&= fT \\ \sum_{i \in \lambda^{-1}(4)} \gamma_i^f &= 3\left(f\frac{T}{3}\right) &&= fT \\ \sum_{i \in \lambda^{-1}(5)} \gamma_i^f &= 3\left(f\frac{T}{3}\right) &&= fT \\ \sum_{i \in \lambda^{-1}(6)} \gamma_i^f &= 3\left(f\frac{T}{3}\right) &&= fT \end{aligned}$$

The same argument is valid for all group permutations of the solution (A.3). The largest objective function from these $m!$ solutions in the form of (A.3) is:

$$\begin{aligned} V_{(\text{A.3})} &= \Big(\underbrace{\Big(\Big(\sum_{i:\lambda(i)=1} \Big(0 + \frac{m-1}{T} r_i\Big)\Big)^{\mu}}_{\text{for group } m} + \underbrace{\Big(\sum_{i:\lambda(i)=c'} 1\Big)^{\mu}}_{\text{for the group in which } \overline{n}+1 \text{ is located}} \Big)^{1/\mu} \\ &\leq \Big(\Big(T \frac{m-1}{T} \Big)^{\mu} + 2^{\mu} \Big)^{1/\mu} \\ &= \Big(\Big(m-1 \Big)^{\mu} + 2^{\mu} \Big)^{1/\mu}. \end{aligned}$$

We now show that there exists n', such that for all $n > n'$, we have $\overline{r} B_{\mu} > V_{(\text{A.3})}$:

$$\begin{aligned} \overline{r} B_{\mu} &= \overline{r} \left(1 + \sum_{h=1}^{m-1} \left(\frac{h}{T} \right)^{\mu} \right)^{1/\mu} \\ &= \left(\left(\frac{T}{m} \right)^{\mu} + \frac{1}{m^{\mu}} \sum_{h=1}^{m-1} h^{\mu} \right)^{1/\mu} \\ &> \left(\left(\frac{T}{m} \right)^{\mu} + \frac{1}{m^{\mu}} (m + m^{\mu} - 2) \right)^{1/\mu} \\ &> \left(\left(\frac{\overline{n}}{m} \right)^{\mu} + \frac{1}{m^{\mu}} (m + m^{\mu} - 2) \right)^{1/\mu} \end{aligned}$$

Hence, n' is obtained from the following inequality:

$$(m-1)^{\mu} + 2^{\mu} < \left(\frac{n-2m}{m} \right)^{\mu} + \frac{1}{m^{\mu}} (m + m^{\mu} - 2).$$

□

Proposition 3

Proof. Consider problem (12) and distinguish the following cases:

(i) When $x_{i,c} + x_{j,c} < 2$, then the optimal value of problem (11) (as well as the one of its dual) is zero. Therefore, a solution of problem (12) is obtained from the following system:

$$\begin{cases} x_{i,c} \overline{\zeta}^{c}_{i,j} + x_{j,c} \underline{\zeta}^{c}_{i,j} + (1 - x_{i,c} - x_{j,c}) \zeta^{c}_{i,j} = 0 \\ \overline{\zeta}^{c}_{i,j} + \underline{\zeta}^{c}_{i,j} - \zeta^{c}_{i,j} \geq \alpha_{i,j} \\ \overline{\zeta}^{c}_{i,j}, \underline{\zeta}^{c}_{i,j}, \zeta^{c}_{i,j} \geq 0. \end{cases}$$

This system admits an infinite amount of solutions. The following is a particular solution:

$$\begin{cases} \overline{\zeta}^{c}_{i,j} = \alpha_{i,j}(1 - x_{i,c}), \quad \underline{\zeta}^{c}_{i,j} = \alpha_{i,j}(1 - x_{j,c}), \quad \zeta^{c}_{i,j} = 0 & \text{if } x_{i,c} + x_{j,c} = 1 \\ \overline{\zeta}^{c}_{i} = \dfrac{\alpha_{i,j}}{2}, \quad \underline{\zeta}^{c}_{j} = \dfrac{\alpha_{i,j}}{2}, \quad \zeta^{c}_{i,j} = 0 & \text{if } x_{i,c} + x_{j,c} = 0. \end{cases}$$

(ii) When $x_{i,c} + x_{j,c} = 2$, then the optimal value of problem (11) (as well as the one of its dual) is $\alpha_{i,j}$. Therefore, a solution of problem (12) is obtained from the following system:

$$\begin{cases} x_{i,c}\overline{\zeta}^c_{i,j} + x_{j,c}\underline{\zeta}^c_{i,j} + (1 - x_{i,c} - x_{j,c})\zeta^c_{i,j} = \alpha_{i,j} \\ \overline{\zeta}^c_{i,j} + \underline{\zeta}^c_{i,j} - \zeta^c_{i,j} \geq \alpha_{i,j} \\ \overline{\zeta}^c_{i,j}, \underline{\zeta}^c_{i,j}, \zeta^c_{i,j} \geq 0 \end{cases}$$

Hence, three solutions can be directly identified:

$$\begin{array}{llll} \text{either} & \overline{\zeta}^c_i = \alpha_{i,j}, & \underline{\zeta}^c_j = 0, & \zeta^c_{i,j} = 0, \\ \text{or} & \overline{\zeta}^c_i = 0, & \underline{\zeta}^c_j = \alpha_{i,j}, & \zeta^c_{i,j} = 0, \\ \text{or} & \overline{\zeta}^c_i = \alpha_{i,j}, & \underline{\zeta}^c_j = \alpha_{i,j}, & \zeta^c_{i,j} = \alpha_{i,j}. \end{array}$$

□

Corollary 1

Proof. Let us consider

$$\begin{aligned} \mathrm{UB}(v) &= \sum_{(i,j)\in\mathcal{E}} \left(x_{i,c}\overline{\zeta}^c_{i,j}(v) + x_{j,c}\underline{\zeta}^c_{i,j}(v) + (1 - x_{i,c} - x_{j,c})\zeta^c_{i,j}(v) \right) \\ &= \sum_{i\in\mathcal{N}} x_{i,c} \sum_{j\in\mathcal{N}} \overline{\zeta}^c_{i,j}(v) + \sum_{j\in\mathcal{N}} x_{j,c} \sum_{i\in\mathcal{N}} \underline{\zeta}^c_{i,j}(v) + \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} (1 - x_{i,c} - x_{j,c})\zeta^c_{i,j}(v) \\ &= \sum_{i\in\mathcal{N}} x_{i,c} \sum_{j\in\mathcal{N}} \overline{\zeta}^c_{i,j}(v) + \sum_{i\in\mathcal{N}} x_{i,c} \sum_{j\in\mathcal{N}} \underline{\zeta}^c_{j,i}(v) + \sum_{(i,j)\in\mathcal{E}} \zeta^c_{i,j}(v) \\ &\quad - \sum_{i\in\mathcal{N}} x_{i,c} \sum_{j\in\mathcal{N}} \zeta^c_{i,j}(v) - \sum_{i\in\mathcal{N}} x_{i,c} \sum_{j\in\mathcal{N}} \zeta^c_{j,i}(v) \\ &= \sum_{i\in\mathcal{N}} x_{i,c} \sum_{j\in\mathcal{N}} (\overline{\zeta}^c_{i,j}(v) - \zeta^c_{i,j}(v)) + \sum_{i\in\mathcal{N}} x_{i,c} \sum_{j\in\mathcal{N}} (\underline{\zeta}^c_{j,i}(v) - \zeta^c_{j,i}(v)) + \sum_{(i,j)\in\mathcal{E}} \zeta^c_{i,j}(v) \\ &= \tilde{\alpha}_c(v) + \sum_{i\in\mathcal{N}} x_{i,c}(\tilde{\alpha}^0_{i,c}(v) + \tilde{\alpha}^1_{i,c}(v)) \end{aligned}$$

where

$$\tilde{\alpha}^0_{i,c}(v) = \sum_{j\in\mathcal{N}} (\overline{\zeta}^c_{i,j}(v) - \zeta^c_{i,j}(v)), \quad \tilde{\alpha}^1_{i,c}(v) = \sum_{i\in\mathcal{N}} (\underline{\zeta}^c_{i,j}(v) - \zeta^c_{i,j}(v)), \quad \tilde{\alpha}_c(v) = \sum_{(i,j)\in\mathcal{E}} \zeta^c_{i,j}(v)$$

By Proposition 3, we have

$$\tilde{\alpha}^0_{i,c}(v) = (1 - x_{i,c}(v)) \sum_{j\in\mathcal{N}} \alpha_{i,j}, \quad \tilde{\alpha}^1_{i,c}(v) = (1 - x_{i,c}(v)) \sum_{j\in\mathcal{N}} \alpha_{j,i}, \quad \tilde{\alpha}_c(v) = \sum_{(i,j)\in\mathcal{E}} \alpha_{i,j} x_{i,c}(v) x_{j,c}(v).$$

Hence, we have

$$\mathrm{UB}(v) = \sum_{(i,j)\in\mathcal{E}} \alpha_{i,j} x_{i,c}(v) x_{j,c}(v) + \sum_{i\in\mathcal{N}} x_{i,c}(1 - x_{c,i}(v)) \left(\sum_{j\in\mathcal{N}} \alpha_{i,j} + \sum_{j\in\mathcal{N}} \alpha_{j,i} \right).$$

□

Proposition 4

Proof. Let $\texttt{CH}(\mathcal{X})$ be the convex hull of $\mathcal{X}$. A core point is a point $\boldsymbol{y}$ is a point in the relative interior of the $\texttt{CH}(\mathcal{X})$ (i.e., the largest subspace for which $\texttt{CH}(\mathcal{X})$ is full dimensional). For any point $\boldsymbol{x}_0 \in \mathcal{X}$, a core point can be constructed as $\tilde{\boldsymbol{x}} = \iota \boldsymbol{x}_0 + (1-\iota)\boldsymbol{y}$, for any $\iota \in (0,1)$, where $\boldsymbol{x}$ is defined as

$$y_{i,c} = \frac{1}{n}\left(\tilde{\iota}\tau_{\max} + (1-\tilde{\iota})\tau_{\min}\right), \qquad \text{and} \qquad \tilde{\iota} = \frac{\dfrac{n}{m} - \tau_{\min}}{\tau_{\max} - \tau_{\min}}.$$

Note that if $\tau_{\min} \leq n/m \leq \tau_{\max}$ then $\iota \in (0,1)$. To show that $\boldsymbol{y}$ is in the relative interior of $\mathcal{X}$, we compute constraints (7a)-(7e):

$$\begin{cases} \displaystyle\sum_{c\in\mathcal{M}} y_{i,c} = \frac{m}{n}\left(\tilde{\iota}\tau_{\max} + (1-\tilde{\iota})\tau_{\min}\right) = 1 & i \in \mathcal{N} \quad \text{(A.4a)} \\ \displaystyle\sum_{i\in\mathcal{N}} y_{i,c} = \left(\tilde{\iota}\tau_{\max} + (1-\tilde{\iota})\tau_{\min}\right) > \tau^{\min} & c \in \mathcal{M} \quad \text{(A.4b)} \\ \displaystyle\sum_{i\in\mathcal{N}} y_{i,c} = \left(\tilde{\iota}\tau_{\max} + (1-\tilde{\iota})\tau_{\min}\right) < \tau^{\max} & c \in \mathcal{M} \quad \text{(A.4c)} \\ \displaystyle\sum_{i\in\mathcal{N}} \gamma_i^f (y_{i,c} - y_{i,c'}) = 0 < \theta_f & f \in \mathcal{F},\ c, c' \in \mathcal{M} \quad \text{(A.4d)} \\ y_{i,c} \in (0,1) & i \in \mathcal{N},\ c \in \mathcal{M} \quad \text{(A.4e)} \end{cases}$$

□

Therefore, equality constraints are satisfied and inequality constraints are strictly satisfied. Further, if $\boldsymbol{x}_0 \in \mathcal{X}$ and $\boldsymbol{y}$ is in the relative interior of $\texttt{CH}(\mathcal{X})$, then for $\iota \in (0,1)$ and for any point $\boldsymbol{x}_0 \in \mathcal{X}$, the point $\tilde{\boldsymbol{x}} = \iota \boldsymbol{x}^{(0)} + (1-\iota)\boldsymbol{y}$ is also in the relative interior of $\texttt{CH}(\mathcal{X})$.

Proposition 5

Proof. To characterize $\overline{\zeta}_{i,j}^c$, $\underline{\zeta}_{i,j}^c$ and $\zeta_{i,j}^c$ for different values of $\boldsymbol{y}$ and $\boldsymbol{x}^{(0)}$, we distinguish hereafter the different cases, where cases (i)-(iv) refer to $x_{i,c} + x_{i,c} = 2$, whereas (v)-(viii) refers to $x_{i,c} + x_{i,c} < 1$.

(i) Case $\tilde{x}_{i,c} \leq \tilde{x}_{j,c} \leq 1 - \tilde{x}_{i,c} - \tilde{x}_{j,c}$ and $x_{i,c} = x_{i,c} = 1$. In this case, $\alpha_{i,j} x_{i,c} x_{j,c} = \alpha_{i,j}$, so that we re-write problem (16) as

$$\begin{cases} \min \quad \tilde{x}_{i,c}\overline{\zeta}_{i,j}^c + \tilde{x}_{j,c}\underline{\zeta}_{i,j}^c + (1 - \tilde{x}_{i,c} - \tilde{x}_{j,c})\zeta_{i,j}^c & \text{(A.5a)} \\ \text{subj. to } \overline{\zeta}_{i,j}^c + \underline{\zeta}_{i,j}^c - \zeta_{i,j}^c \geq \alpha_{i,j} & \text{(A.5b)} \\ \overline{\zeta}_{i,j}^c + \underline{\zeta}_{i,j}^c - \alpha_{i,j} = \zeta_{i,j}^c & \text{(A.5c)} \\ \overline{\zeta}_{i,j}^c, \underline{\zeta}_{i,j}^c, \zeta_{i,j}^c \geq 0 & \text{(A.5d)} \end{cases}$$

which implies

$$\begin{cases} \min \quad (1 - \tilde{x}_{j,c})\overline{\zeta}_{i,j}^c + (1 - \tilde{x}_{i,c})\underline{\zeta}_{i,j}^c & \text{(A.6a)} \\ \text{subj. to} \quad \overline{\zeta}_{i,j}^c + \underline{\zeta}_{i,j}^c \geq \alpha_{i,j} & \text{(A.6b)} \\ \overline{\zeta}_{i,j}^c, \underline{\zeta}_{i,j}^c \geq 0 & \text{(A.6c)} \end{cases}$$

Therefore, $\overline{\zeta}_{i,j}^c = (\alpha_{i,j})^+$, $\underline{\zeta}_{i,j}^c = 0$, and $\zeta_{i,j}^c = 0$.

(ii) Case $\tilde{x}_{j,c} \leq \tilde{x}_{i,c} \leq 1 - \tilde{x}_{i,c} - \tilde{x}_{j,c}$ and $x_{i,c} = x_{i,c} = 1$. We apply a symmetric reasoning and obtain $\overline{\zeta}^c_{i,j} = 0$, $\underline{\zeta}^c_{i,j} = (\alpha_{i,j})^+$, and $\zeta^c_{i,j} = 0$.

(iii) Case $\tilde{x}_{i,c} \leq 1 - \tilde{x}_{i,c} - \tilde{x}_{j,c} \leq \tilde{x}_{j,c}$ and $x_{i,c} = x_{i,c} = 1$. In this case, $\alpha_{i,j} x_{i,c} x_{j,c} = \alpha_{i,j}$, so that we re-write problem (16) as (A.5), which implies (A.6). Since $\tilde{x}_{j,c} \geq \tilde{x}_{i,c}$, the cost of $\overline{\zeta}^c_{i,j}$ in (A.6) is less or equal to the one of $\underline{\zeta}^c_{i,j}$. Therefore, the solution becomes $\overline{\zeta}^c_{i,j} = (\alpha_{i,j})^+$, $\underline{\zeta}^c_{i,j} = 0$, and $\zeta^c_{i,j} = 0$.

(iv) Case $\tilde{x}_{j,c} \leq 1 - \tilde{x}_{i,c} - \tilde{x}_{j,c} \leq \tilde{x}_{i,c}$ and $x_{i,c} = x_{i,c} = 1$. In this case, $\alpha_{i,j} x_{i,c} x_{j,c} = \alpha_{i,j}$, so that we re-write problem (16) as (A.5), which implies (A.6). Since $\tilde{x}_{i,c} \geq \tilde{x}_{j,c}$, the cost of $\overline{\zeta}^c_{i,j}$ in (A.6) is greater or equal to the one of $\underline{\zeta}^c_{i,j}$. Therefore, the solution becomes $\overline{\zeta}^c_{i,j} = 0$, $\underline{\zeta}^c_{i,j} = (\alpha_{i,j})^+$, and $\zeta^c_{i,j} = 0$.

(v) Case $\tilde{x}_{i,c} \leq 1 - \tilde{x}_{i,c} - \tilde{x}_{j,c}$ and $x_{i,c} = 0$ and $x_{j,c} = 1$. In this case, $\alpha_{i,j} x_{i,c} x_{j,c} = 0$, so that we re-write problem (16) as

$$\begin{cases} \min \quad \tilde{x}_{i,c}\overline{\zeta}^c_{i,j} + (1 - \tilde{x}_{i,c} - \tilde{x}_{j,c})\zeta^c_{i,j} & \text{(A.7a)} \\ \text{subj. to } \overline{\zeta}^c_{i,j} - \zeta^c_{i,j} \geq \alpha_{i,j} & \text{(A.7b)} \\ \overline{\zeta}^c_{i,j}, \zeta^c_{i,j} \geq 0 & \text{(A.7c)} \end{cases}$$

Therefore, $\overline{\zeta}^c_{i,j} = (\alpha_{i,j})^+$, $\underline{\zeta}^c_{i,j} = 0$, and $\zeta^c_{i,j} = 0$.

(vi) Case $\tilde{x}_{j,c} \leq 1 - \tilde{x}_{i,c} - \tilde{x}_{j,c}$ and $x_{i,c} = 0$ and $x_{j,c} = 1$. In this case, $\alpha_{i,j} x_{i,c} x_{j,c} = 0$, so that we re-write problem (16) as

$$\begin{cases} \min \quad \tilde{x}_{j,c}\underline{\zeta}^c_{i,j} + (1 - \tilde{x}_{i,c} - \tilde{x}_{j,c})\zeta^c_{i,j} & \text{(A.8a)} \\ \text{subj. to } \underline{\zeta}^c_{i,j} - \zeta^c_{i,j} \geq \alpha_{i,j} & \text{(A.8b)} \\ \underline{\zeta}^c_{i,j}, \zeta^c_{i,j} \geq 0 & \text{(A.8c)} \end{cases}$$

Therefore, $\overline{\zeta}^c_{i,j} = 0$, $\underline{\zeta}^c_{i,j} = (\alpha_{i,j})^+$, and $\zeta^c_{i,j} = 0$.

(vii) Case $\tilde{x}_{i,c} > 1 - \tilde{x}_{i,c} - \tilde{x}_{j,c}$ and $x_{i,c} = 0$ and $x_{j,c} = 1$. In this case, $\alpha_{i,j} x_{i,c} x_{j,c} = 0$, so that we re-write problem (16) as

$$\begin{cases} \min \quad \tilde{x}_{i,c}\overline{\zeta}^c_{i,j} + (1 - \tilde{x}_{i,c} - \tilde{x}_{j,c})\zeta^c_{i,j} & \text{(A.9a)} \\ \text{subj. to } \overline{\zeta}^c_{i,j} - \zeta^c_{i,j} \geq \alpha_{i,j} & \text{(A.9b)} \\ \overline{\zeta}^c_{i,j}, \zeta^c_{i,j} \geq 0 & \text{(A.9c)} \end{cases}$$

Therefore, $\overline{\zeta}^c_{i,j} = 0$, $\underline{\zeta}^c_{i,j} = 0$, and $\zeta^c_{i,j} = (-\alpha_{i,j})^+$.

(viii) Case $\tilde{x}_{j,c} > 1 - \tilde{x}_{i,c} - \tilde{x}_{j,c}$ and $x_{i,c} = 0$ and $x_{j,c} = 1$. In this case, $\alpha_{i,j} x_{i,c} x_{j,c} = 0$, so that we re-write problem (16) as

$$\begin{cases} \min \quad \tilde{x}_{j,c}\underline{\zeta}^c_{i,j} + (1 - \tilde{x}_{i,c} - \tilde{x}_{j,c})\zeta^c_{i,j} & \text{(A.10a)} \\ \text{subj. to } \underline{\zeta}^c_{i,j} - \zeta^c_{i,j} \geq \alpha_{i,j} & \text{(A.10b)} \\ \underline{\zeta}^c_{i,j}, \zeta^c_{i,j} \geq 0 & \text{(A.10c)} \end{cases}$$

Therefore, $\overline{\zeta}^c_{i,j} = 0$, $\underline{\zeta}^c_{i,j} = 0$, and $\zeta^c_{i,j} = (-\alpha_{i,j})^+$.

□

B : Details of the illustrative example

We deduce formulations (18) and (19), corresponding to the simplified illustrative example of problems (2) and (3) with $n = 2$, $m = 2$, $\tau_{\min} = 1$, $\tau_{\max} = 2$ and homogeneity constraints, as introduced in Section (6). The following respective formulations of problems (2) and (3) involve 6 binary variables, 4 inequality constraints and 3 equality constraints:

$$\left\{\begin{array}{ll} \max & \eta_1 + \eta_2 \\ \\ \text{subj. to} & \eta_1 \leq \begin{pmatrix} x_{1,1} \\ x_{2,1} \\ x_{3,1} \end{pmatrix}^\top \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_{1,1} \\ x_{2,1} \\ x_{3,1} \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} x_{1,1} \\ x_{2,1} \\ x_{3,1} \end{pmatrix} \\ \\ & \eta_2 \leq \begin{pmatrix} x_{1,2} \\ x_{2,2} \\ x_{3,2} \end{pmatrix}^\top \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_{1,2} \\ x_{2,2} \\ x_{3,2} \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} x_{1,2} \\ x_{2,2} \\ x_{3,2} \end{pmatrix} \end{array}\right. \tag{B.1}$$

and

$$\left\{\begin{array}{ll} \max & \eta \\ \\ \text{subj. to} & \eta \leq \begin{pmatrix} x_{1,1} \\ x_{2,1} \\ x_{3,1} \end{pmatrix}^\top \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_{1,1} \\ x_{2,1} \\ x_{3,1} \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} x_{1,1} \\ x_{2,1} \\ x_{3,1} \end{pmatrix}. \\ \\ & \eta \leq \begin{pmatrix} x_{1,2} \\ x_{2,2} \\ x_{3,2} \end{pmatrix}^\top \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_{1,2} \\ x_{2,2} \\ x_{3,2} \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} x_{1,2} \\ x_{2,2} \\ x_{3,2} \end{pmatrix}. \end{array}\right. \tag{B.2}$$

with the inclusion of the following constraints:

$$\left\{\begin{array}{l} x_{1,1} + x_{1,2} = 1 \\ x_{2,1} + x_{2,2} = 1 \\ x_{3,1} + x_{3,2} = 1 \\ \\ x_{1,1} + x_{2,1} + x_{3,1} \geq 1 \\ x_{1,2} + x_{2,2} + x_{3,2} \geq 1 \\ \\ x_{1,1},\ x_{2,1},\ x_{3,1},\ x_{1,2},\ x_{2,2},\ x_{3,2} \ \text{ binary.} \end{array}\right.$$

Using the equality constraints (7a), we remove variables $x_{1,1}$, $x_{2,2}$ and $x_{3,2}$:

$$\left\{\begin{array}{l} x_{1,1} = 1 - x_{1,2} \\ x_{2,2} = 1 - x_{2,1} \\ x_{3,2} = 1 - x_{3,1}. \end{array}\right.$$

Replacing it into problems (B.1) and (B.2), with a few algebrical simplifications, we obtain:

$$\begin{cases} \max & \eta_1 + \eta_2 \\ \text{subj. to} & \eta_1 \leq -2x_{2,1}x_{3,1} + (1 - x_{1,2}) \\ & \eta_2 \leq -2(1 - x_{2,1})(1 - x_{3,1}) + (1 - x_{2,1}) + (1 - x_{3,1}) \\ & (1 - x_{1,2}) + x_{2,1} + x_{3,1} \geq 1 \\ & x_{1,2} + (1 - x_{2,1}) + (1 - x_{3,1}) \geq 1 \\ & x_{2,1},\ x_{3,1},\ x_{1,2} \ \text{ binary.} \end{cases} \tag{B.3}$$

and

$$\begin{cases} \max & \eta \\ \text{subj. to} & \eta \leq -2x_{2,1}x_{3,1} + (1 - x_{1,2}) \\ & \eta \leq -2(1 - x_{2,1})(1 - x_{3,1}) + (1 - x_{2,1}) + (1 - x_{3,1}) \\ & (1 - x_{1,2}) + x_{2,1} + x_{3,1} \geq 1 \\ & x_{1,2} + (1 - x_{2,1}) + (1 - x_{3,1}) \geq 1 \\ & x_{2,1},\ x_{3,1},\ x_{1,2} \ \text{ binary.} \end{cases} \tag{B.4}$$

Focusing on constraints

$$\begin{cases} x_{1,2} \leq x_{2,1} + x_{3,1} \\ x_{1,2} + 1 \geq x_{2,1} + x_{3,1} \\ x_{2,1},\ x_{3,1},\ x_{1,2} \ \text{ binary.} \end{cases}$$

we notice that $x_{1,2} = x_{2,1}x_{3,1}$. Therefore, problems (B.4) and (B.4) reduce to (18) and (19), respectively.

The truth table of of $x_{2,1}$ and $x_{3,1}$ presented below showcases a comparative analysis of the values of η_1, η_2, and η derived from problems (29) and (30), against the outcomes η_1', η_2, and η' obtained from problems (18) and (19).

$x_{2,1}$	$x_{3,1}$	η_1	η_1'	η_2	η_2'	η	η'
0	0	1	4	0	2	0	2
0	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1
1	1	-2	-2	0	0	-2	-2

Table B.1: Comparable truth table of $x_{2,1}$ and $x_{3,1}$ showcasing the values of η_1, η_2, and η obtained from problems (18) and (19) against η_1', η_2, and η' obtained from (29) and (30)

C : Instance sizes

Table C.1 reports the problem sizes in terms of qubits of the QUBO reformulations and decision variables and constraints for the MILP reformulations.

n	m	f	# binary var	# continuous var	# constraints	# $Q_{\text{min\_sum}}$	# $Q_{\text{min\_max}}$
21	3	2	63	1323	4065	105	1023
49	10	1	490	24010	72659	730	5578
100	10	2	1000	100000	301280	1620	21674
200	20	2	4000	800000	2404960	6440	86502
300	30	2	9000	2700000	8111040	14460	194526
400	40	2	16000	6400000	19219520	25680	345750

Table C.1: The table is organized from left to right, starting with the number of individuals to be grouped, denoted as n, followed by the number of groups to be formed, m, and the number of features considered by the heterogeneity constraints, f. It also specifies the number of binary variables required for both problems (# binary var), as well as the number of continuous variables (# continuous var) alongside with the total number of constraints (# constraints) needed for both problems. The table concludes with the number of binary variables required for the QUBO reformulations of both problem, represented as $Q_{\text{min\_sum}}$ and $Q_{\text{min\_max}}$, respectively.

D : MILP reformulation of the relaxed R-MN-SAP problem

This appendix provides a MILP reformulation of the relaxed R-MN-SAP problem (4), which is obtained by using a first order approximation of the μ-norm on a collection $\mathcal{T}$ of interpolation seeds with values ℓ_t, for $t \in \mathcal{T}$. It results in a piecewise linear approximation, with the following MILP reformulation:

$$
\begin{cases}
\min \quad \phi & & \text{(D.1a)} \\
\text{s. to} \quad \phi \geq \left(\sum_{c \in \mathcal{M}} (\ell_t)^{\mu} \right)^{1/\mu} + \sum_{c \in \mathcal{M}} (\ell_t)^{\mu-1} \left(m \, (\ell_t)^{\mu-1} \right)^{1/\mu} (\eta_c - \ell_t) & t \in \mathcal{T} & \text{(D.1b)} \\
\eta_c = \kappa - \sum_{i>j} \alpha_{i,j} x_{i,c} x_{j,c} - \sum_{i \in \mathcal{N}} \beta_{i,c} x_{i,c} & c \in \mathcal{M} & \text{(D.1c)} \\
\sum_{c \in \mathcal{M}} x_{i,c} = 1 & i \in \mathcal{N} & \text{(D.1d)} \\
\sum_{i \in \mathcal{N}} x_{i,c} \geq \tau^{\min} & c \in \mathcal{M} & \text{(D.1e)} \\
\sum_{i \in \mathcal{N}} x_{i,c} \leq \tau^{\max} & c \in \mathcal{M} & \text{(D.1f)} \\
\sum_{i \in \mathcal{N}} \gamma_i^f (x_{i,c} - x_{i,c'}) \leq \theta_f & f \in \mathcal{F}, \; c, c' \in \mathcal{M} & \text{(D.1g)} \\
x_{i,c} \;\; \text{binary} & i \in \mathcal{N}, \; c \in \mathcal{M}. & \text{(D.1h)}
\end{cases}
$$