Received: 9 September 2022 | Revised: 21 March 2023 | Accepted: 9 June 2023

DOI: 10.1002/mma.9489

RESEARCH ARTICLE

WILEY

Solving the Kemeny ranking aggregation problem with quantum optimization algorithms

Elías F. Combarro[1] | **Raúl Pérez-Fernández**[2] | **José Ranilla**[1] | **Bernard De Baets**[3]

[1]Quantum and High Performance Computing Group, Computer Science Department, University of Oviedo, Oviedo, Spain

[2]Department of Statistics and O.R. and Mathematics Didactics, University of Oviedo, Oviedo, Spain

[3]KERMIT, Department of Data Analysis and Mathematical Modelling, Ghent University, Ghent, Belgium

Correspondence
Elías F. Combarro, Quantum and High Performance Computing Group, Computer Science Department, University of Oviedo, Oviedo, Spain.
Email: efernandezca@uniovi.es

Communicated by: J. Vigo-Aguiar

Present address
Elías F. Combarro, Office 4.1, Faculty of Geology, C. Jesús Arias de Velasco, Oviedo, Asturias, Spain

Funding information
MCIN/AEI/10.13039/501100011033, Grant/Award Number: PID2020-119082RB-C22; Gobierno del Principado de Asturias, Grant/Award Number: AYUD/2021/50994; Spanish Ministerio de Economía y Competitivad, Grant/Award Number: TIN2017-87600-P; Ministry of Economic Affairs and Digital Transformation of the Spanish Government and the European Union through the Recovery European Union through the Recovery, Transformation and Resilience Plan - NextGenerationEU, Grant/Award Number: QUANTUMSPAIN; Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" program

The aim of a ranking aggregation problem is to combine several rankings into a single one that best represents them. A common method for solving this problem is due to Kemeny and selects as the aggregated ranking the one that minimizes the sum of the Kendall distances to the rankings to be aggregated. Unfortunately, the identification of the said ranking—called the Kemeny ranking—is known to be a computationally expensive task. In this paper, we study different ways of computing the Kemeny ranking with quantum optimization algorithms, and in particular, we provide some alternative formulations for the search for the Kemeny ranking as an optimization problem. To the best of our knowledge, this is the first time that this problem is addressed with quantum techniques. We propose four different ways of formulating the problem, one novel to this work. Two different quantum optimization algorithms—Quantum Approximate Optimization Algorithm and Quantum Adiabatic Computing—are used to evaluate each of the different formulations. The experimental results show that the choice of the formulation plays a big role on the performance of the quantum optimization algorithms.

KEYWORDS
QAOA, quantum annealing, quantum computing, ranking aggregation

MSC CLASSIFICATION
68W25

1 | INTRODUCTION

The aim of a ranking aggregation problem is to combine several rankings into a single one. This problem typically arises in the context of Politics and Elections; therefore, it has mostly been addressed by the field of Social Choice Theory [1]. However, since the need for aggregating rankings arises in almost all fields of application, this problem has also been addressed in fields as varied as Economics [2], Medicine [3], and Computer Science [4]. Even though the ranking aggregation problem is conceptually easy, the insights on the problem are quite rich, typically resulting in many paradoxes and interesting situations. For instance, Arrow [5] pointed out in his now-called Arrow's impossibility theorem that there exists no method for solving the ranking aggregation problem that fulfills a basic set of properties (namely, nondictatorship, unanimity, and independence of irrelevant alternatives). Of course, this is not to say that there do not exist methods for solving the ranking aggregation problem that are better than others. For instance, the Kemeny method [6], which can be understood as solving a kind a maximum likelihood estimation problem [7], is known to fulfill many intuitive properties [8].

The main drawback of the Kemeny method is its computational inefficiency since computing the Kemeny ranking is known to be an NP-hard problem [9]. For this reason, research has been carried out on the implementation of efficient methods for its computation. For instance, Davenport and Kalagnanam have opted for branch-and-bound methods reducing the search space from all possible rankings to only some carefully selected rankings [10]. These bounds were improved in Conitzer et al. [11] by considering integer linear programming. A similar direction was followed by Azzini and Munda [12], who presented new bounds related to constraints based on the voting matrix and, additionally, introduced a heuristic method for the computation of the Kemeny ranking that allows to deal with rankings of a larger number of objects but does not guarantee the obtained solution to be optimal. As discussed by Schalekamp and van Zuylen [13], several other heuristic methods for the computation of the Kemeny ranking with varying performance guarantees and computational complexity have been proposed. Here, we study this problem from the perspective of Quantum Computing [14], by exploring several different approaches for finding the Kemeny ranking with quantum optimization algorithms. Since the developed algorithms do not rely on the assumption of the input relations being rankings, we also consider here a slight generalization of the ranking aggregation problem in which the input relations to be aggregated are not required to be total. Interestingly, this setting, in which more general types of binary relations are considered for the input rankings, has already attracted the attention of the research community [15, 16], especially for the case of top-k lists [17]. On the contrary, we will not consider here the even more general setting of the aggregation of binary relations [18, 19], in which the output binary relation is not required to be a ranking. This is because altering the type of output binary relation will result in an optimization problem with different constraints, and actually, the main goal of this paper is to explore the effect of considering alternative representations of the constraints associated with a ranking when considering different quantum algorithms.

More concretely, we consider two different kinds of quantum optimization methods: quantum annealing [20] and the quantum approximate optimization algorithm (QAOA) [21]. Although these two methods can be run on actual quantum hardware available today, the technology is still in its infancy and demonstrable advantage of quantum over classical computing has only been achieved for artificial, academic problems [22] and at a high cost. For this reason, we do not expect quantum computers to offer results on the ranking aggregation problem (or other combinatorial optimization problems) that beat classical algorithms yet. However, it is important to prepare the ground for the time when bigger, more resilient to noise quantum computers will become available, by exploring which formulations of combinatorial optimization problems are more suitable for each task. This is a very active area of research, and for instance, in the last few years, several authors have studied the merits of different quantum formulations of the Travelling Salesperson Problem and its variants [23–25], of graph-coloring problems [26, 27], of workflow scheduling problems [28, 29], and of vehicle routing problems [30, 31].

In this paper, we conduct a study of four different formulations of the Kemeny ranking aggregation problem. To the best of our knowledge, this is the first time that this problem is approached with quantum techniques. We evaluate the merits of each of these formulations, and we test them in both quantum simulators and real quantum computers. Our results show that one of these four formulations is clearly superior to the others on all the metrics that we have considered.

The remainder of this paper is structured as follows. Sections 2 and 3 are preliminary sections needed for the proper understanding of the present paper. More precisely, the former is devoted to the formalization of the ranking aggregation problem, whereas the latter presents the basics of quantum algorithms in the context of combinatorial optimization. Section 4 presents different formulations of the ranking aggregation problem as a Quadratic Unconstrained Binary Optimization (QUBO) problem, which can be solved by means of quantum algorithms. Experimental results concerning different implementations of quantum algorithms for the different formulations of the ranking aggregation problem are presented in Section 5. We end with some conclusions and ideas for further research in Section 6.

2 | FORMALIZATION OF THE RANKING AGGREGATION PROBLEM

In this work, we will consider a ranking aggregation problem. Namely, we are given a set of objects $O = \{1, \dots, n\}$ and a collection of m partial orders $\{P_1, \dots, P_m\}$ on O, where $(i_1, i_2) \in P_i$ means that i_1 is greater than or equal to i_2 according to the i-th partial order. The goal is to find the ranking T on O (i.e., a total order relation on O) that best represents the partial orders $\{P_1, \dots, P_m\}$. Note that this setting slightly generalizes the more classical setting in which all the partial orders $\{P_1, \dots, P_m\}$ are also rankings. More specifically, we are here interested in a particular type of ranking aggregation problem in which the ranking T is required to minimize the number of pairwise incompatibilities with the given partial orders, that is, we want to find a total order T minimizing the cost

$$\sum_{i,j=1}^{n} c_{ij} T_{ij},$$

where T_{ij} represents the characteristic mapping of T ($T_{ij} = 1$ if $(i,j) \in T$ and $T_{ij} = 0$ otherwise) and $C = (c_{ij})_{i,j=1}^{n}$ is referred to as the cost matrix of the problem, defined as $c_{ii} = 0$ for any i and whenever $i \neq j$, c_{ij} equals the number of partial orders P_k for which $(j,i) \in P_k$. Note that C only depends on the partial orders $P_1, \dots, P_m$ and, actually, needs to be defined differently if T is not required to be a ranking but another type of binary relation (see Barthelemy and Monjardet [18] for a discussion on the more general case of the aggregation of binary relations). The ranking that minimizes the cost is known as the Kemeny ranking (or Kemeny rankings, in case the minimizer is not unique). The search for the Kemeny ranking is known to be NP-hard, and several heuristic and approximate algorithms have been proposed in order to solve it [32].

Fortunately, there exist a couple of scenarios under which the computation of the Kemeny ranking is straightforward. Firstly, the easiest case is that of the unanimous voting scenario, in which all rankings are total and the same. Obviously, the Kemeny ranking under the unanimous voting scenario is the very same unanimous ranking. Secondly, given the cost matrix C, we can define a binary relation, called the simple majority relation M [33], defined by $M_{ij} = 1$ if $c_{ij} \leq c_{ji}$ and $M_{ij} = 0$ otherwise. If m is odd and all partial orders are total, then M is assured to be a tournament [34], that is, an antisymmetric and total binary relation. However, the simple majority relation is not transitive in general, and it can even have cycles. Interestingly, in case M is transitive (henceforth called the transitive tournament scenario), it is a total order and it is the solution of the Kemeny ranking aggregation problem.

In order to better understand the experiments that we present in Section 5, let us give an illustrative example. Consider the set of objects $O = \{1, 2, 3\}$, on which we have the following five partial orders (only strict parts are listed for brevity):

$$\begin{aligned}
P_1 &= \{1 > 2, 1 > 3\},\\
P_2 &= \{2 > 1, 1 > 3, 2 > 3\},\\
P_3 &= \{1 > 3\},\\
P_4 &= \{3 > 1, 3 > 2\},\\
P_5 &= \{3 > 1, 1 > 2, 3 > 2\}.
\end{aligned}$$

The cost matrix of the problem is

$$C = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 0 & 2 \\ 3 & 1 & 0 \end{pmatrix}.$$

The costs of the six possible total orders given this cost matrix are shown in Table 1. As can be seen, in this case, the optimal solution is 1 > 3 > 2 with a cost of 4. Note that the simple majority matrix is

$$M = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix},$$

which is a transitive tournament corresponding to the total order 1 > 3 > 2, which is indeed the optimal solution to the problem.

TABLE 1 Total orders and their costs.

Order	Cost
$1 > 2 > 3$	5
$1 > 3 > 2$	4
$2 > 1 > 3$	6
$2 > 3 > 1$	7
$3 > 1 > 2$	5
$3 > 2 > 1$	6

3 | QUANTUM ALGORITHMS FOR COMBINATORIAL OPTIMIZATION

The usual approach to solving combinatorial optimization problems with quantum algorithms involves reformulating the problems in a way that quantum computers can natively tackle them. One of the most popular ways of doing this is by recasting the optimization problem as an instance of finding the ground state of a Hamiltonian.

Finding the ground state of a given Hamiltonian is a problem that has received much attention from the quantum computing community [20, 21, 35, 36]. To state it formally, we need to define some concepts and introduce some notations. A Hermitian (or self-adjoint) matrix is a complex square matrix H that is equal to its conjugate transpose $H^\dagger$ (note that if all the entries of H are real, then H is Hermitian if and only if it is symmetric). In the field of quantum computing, the Dirac notation is used to represent vectors, and thus, $|x\rangle$ will represent a column vector and $\langle x|$ its conjugate transpose (which is a row vector). Notice that if H is Hermitian, then $\langle x| H |x\rangle$ is a real value because $(\langle x| H |x\rangle)^\dagger = \langle x| H^\dagger |x\rangle = \langle x| H |x\rangle$. The problem we are interested in can be formulated as follows: given an Hermitian matrix H, find $|x\rangle$ such that $\langle x| H |x\rangle$ is minimized.

This problem is of great importance in quantum physics and quantum chemistry. H, called a Hamiltonian or an observable, usually represents some quantity of interest in a physical or chemical system, and $\langle x| H |x\rangle$ is the expected value of that quantity when we measure a system whose state is given by the vector $|x\rangle$. It has been proposed that quantum computers may be especially suitable for solving this kind of problem, for instance, by using algorithms such as the Variational Quantum Eigensolver (VQE) [35]. Additionally, many combinatorial optimization problems can be reduced to finding the ground state of a certain Hamiltonian, something that has spurred a lot of research on the possibility of using quantum computers to find good approximate solutions to hard optimization problems.

Indeed, if we encode an optimization problem as finding the ground state (we will explain some techniques for doing so in the following sections) of a Hamiltonian H, then we can use several different quantum computing approaches to obtain an approximate solution to our problem. One of the most popular ones is the use of Quantum Adiabatic Computing [20], in which we consider a Hamiltonian H_f whose ground states encode solutions to the combinatorial problem we are interested in and another Hamiltonian H_i whose ground state $|\phi_i\rangle$ we can easily find. Then, we consider a time-dependent Hamiltonian $H(t)$ such that $H(0) = H_i$ and $H(T) = H_f$ as, for instance,

$$H(t) = \left(1 - \frac{t}{T}\right) H_i + \frac{t}{T} H_f ,$$

and we use it to make the system evolve through states $|\phi(t)\rangle$ for $t \in [0, T]$. More concretely, at the initial time $t = 0$, the system will be in state $|\phi(0)\rangle = |\phi_i\rangle$, and for $t > 0$, $|\phi(t)\rangle$ will be the solution of the time-dependent Schrödinger equation with Hamiltonian $H(t)$.

The adiabatic theorem [37] assures that if we have a system that is initially in a ground state of H_i (i.e., $|\phi(0)\rangle = |\phi_i\rangle$) and the evolution of the Hamiltonian $H(t)$ acting on the system is slow enough (i.e., if T is big enough), then we will remain in a ground state of $H(t)$ for all $t \in [0, T]$. In particular, $|\phi(T)\rangle$ will be a ground state of $H(T) = H_f$ and, hence, a solution to our problem.

The quantum annealers built by companies such as D-Wave use this kind of approach with several simplifications to make its use feasible in practice. In quantum annealing [20], T is not always guaranteed to be adiabatic, so we will not obtain a ground state of the final Hamiltonian with certainty but only with a certain probability. Also, the Hamiltonian H_f cannot be any Hermitian matrix but one of the form

$$H_f = \sum_{i<j} J_{i,j} Z_i Z_j + \sum_i h_i Z_i ,$$

where the coefficients $J_{i,j}$ and h_i are tunable real numbers, Z_i is the Pauli matrix $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ acting on qubit i, and where Z_iZ_j is the tensor product $I \otimes \cdots \otimes Z \otimes I \otimes \cdots \otimes Z \otimes \cdots \otimes I$ of n two by two matrices, where at positions i and j the factor is Z and the rest of the factors are the identity matrix. This type of matrix is an Ising Hamiltonian, and although it is just a particular case, using only this kind of Hermitian matrix is not very restrictive if we are interested in combinatorial optimization problems, because most of them can be formulated in this way. In fact, it can be proved that finding its ground state is an NP-hard problem [20], so many common optimization problems can be rewritten as an instance of finding the ground state of an Ising Hamiltonian.

In practice, since adiabaticity is not guaranteed, quantum annealers are used several times with the same Hamiltonian and initial state to prepare a final state that it is then measured. Then, the cost of the measured solutions is computed and the best solution is returned as the final result of the algorithm.

As an alternative to quantum annealing, another possibility is to use a discretized version of adiabatic evolution that can be implemented in gate-based quantum computers (such as the ones built by Google or IBM). For instance, in the QAOA [21], we consider the parametrized quantum state

$$|\beta, \gamma\rangle = e^{-i\beta_p H_i} e^{-i\gamma_p H_f} \ \dots \ e^{-i\beta_2 H_i} e^{-i\gamma_2 H_f} e^{-i\beta_1 H_i} e^{-i\gamma_1 H_f} \ |\phi\rangle \ ,$$

where β and γ are real p-tuples (called angles), $p \geq 1$ is an integer parameter, and $|\phi\rangle$ is a state that is easy to prepare. When p approaches ∞, the state obtained with QAOA tends to the one obtained with quantum adiabatic evolution, but even for small values of p, the state $|\beta, \gamma\rangle$ can have a sizeable overlap with the ground states of H_f and lead, upon measurement, to good approximate solutions of our problem [21].

In practice, QAOA is used as a hybrid algorithm in which both classical and quantum computing resources are used. The classical computer selects the initial values for p, β, and γ and asks the quantum computer to prepare and measure the state $|\beta, \gamma\rangle$ several times. The results are used by the classical computer to estimate the energy (cost) of $|\beta, \gamma\rangle$ and, then, to select new values for β and γ in order to try to minimize this energy. This is usually performed with a classical minimization algorithm (see Fernández-Pendás et al. [38], for instance, for more details) until some convergence criteria are met. This will return some optimal values β^* and γ^* for the parameters. Then, the state $|\beta^*, \gamma^*\rangle$ can be prepared and measured in the quantum computer to obtain (approximate) solutions to the optimization problem.

An additional advantage of QAOA is that we are no longer restricted to just QUBO problems, as is the case with quantum annealing, but we can work with higher-order binary optimization (HOBO) problems in which the objective function is a polynomial (of any degree) in the binary variables. For more details on quantum annealing and QAOA, we refer the reader to Combarro and González-Castillo [39], where a full treatment is given starting completely from scratch.

In the following section, we develop several ways of encoding the ranking aggregation problem as an instance of the problem of finding a ground state of a Hamiltonian, so that we can use the kind of quantum algorithms that we have described in this section to solve it.

4 | QUANTUM ALGORITHMS FOR SOLVING THE RANKING AGGREGATION PROBLEM

4.1 | QUBO formulations of the ranking aggregation problem

A usual way of transforming an optimization problem into an instance of finding the ground state of an Ising Hamiltonian (that can then be solved with quantum annealing or with QAOA) is by first formulating it as a QUBO problem. The general form of a QUBO problem is

$$\begin{aligned} &\text{Minimize} \quad \sum_{i,j=1}^{n} q_{ij} x_i x_j \\ &\text{subject to} \quad x_i \in \{0, 1\} \quad i = 1, \dots, n \end{aligned}$$

where q_{ij} are real coefficients. Sometimes it is imposed that $q_{ij} = q_{ji}$, but this is not strictly necessary since we can always define new coefficients $q'_{ij} = \frac{q_{ij}+q_{ji}}{2}$ that satisfy the symmetry condition and do not modify the objective function.

The transformation of a QUBO problem into an Ising Hamiltonian instance is, therefore, straightforward since we only need to consider the Hermitian matrix

$$H = \sum_{i,j=1}^{n} q_{ij} \frac{(I - Z_i)}{2} \frac{(I - Z_j)}{2},$$

which is easily verified to be of the Ising form up to an additive constant.

In practice, however, choosing the right formulation can be of paramount importance in practical applications of quantum optimization techniques. In recent years, several researchers have explored the performance of quantum optimization techniques under different formulations for famous problems such as the Travelling Salesperson Problem [23–25], graph coloring [26, 27], vehicle routing [30, 31], and workflow scheduling [28, 29], showing that there can be significant differences among them.

However, to the best of our knowledge, how to approach the Kemeny ranking aggregation problem remains unexplored to this day. For this reason, in the rest of this section, we will present four different ways of formulating the Kemeny ranking aggregation problem in the QUBO framework. Then, in Section 5, we will test their performance with different quantum optimization techniques using both quantum simulators and real quantum hardware.

4.2 | Formulation via the linear ordering problem (LOP)

Notice that we can formulate the Kemeny ranking aggregation problem as the following 0/1 linear integer program

$$\begin{array}{lrr} \text{Minimize} & \displaystyle\sum_{i,j=1}^{n} c_{ij} T_{ij} & \\ \text{subject to} & T_{ij} \in \{0,1\} & i,j = 1, \ldots, n \\ & T_{ii} = 1 & i = 1, \ldots, n \\ & T_{ij} + T_{ji} = 1 & 1 \leq i < j \leq n \\ & T_{ij} + T_{jk} + T_{ki} \leq 2 & i,j,k = 1, \ldots, n, i < j, i < k, j \neq k. \end{array}$$

The three linear constraints make the binary relation T to be reflexive, antisymmetric and transitive, and hence a ranking. This formulation is sometimes referred to as the LOP [32]. For this reason, we will call this formulation the LOP formulation.

In order to formulate an equivalent QUBO problem, we need to eliminate the constraints by including them as additional penalty terms in the quadratic function to minimize (see Lucas [40], for instance). Obviously, the reflexivity constraints $T_{ii} = 1$ can just be eliminated, and in this way, we may reduce the number of variables to $n^2 - n$. We can do something similar with the antisymmetry constraints: consider just variables of the form T_{ij} with $i < j$ and whenever a variable T_{ij} with $i > j$ appears, replace it by $1 - T_{ji}$. Notice that this will introduce some constant terms in the objective function, but we can simply ignore them because they will not affect the choice of variables that attain the overall minimum value. This simplification reduces the number of variables to $\frac{n^2-n}{2}$.

This leaves only the transitivity constraints to deal with. After the variable transformation, we will have two different cases for these constraints. Since we can always reorder the variables in the constraints and assume without loss of generality that $i < j$ and $i < k$, we will either have

$$T_{ij} + T_{jk} + (1 - T_{ik}) \leq 2$$

when $j < k$ or

$$T_{ij} + (1 - T_{kj}) + (1 - T_{ik}) \leq 2$$

when $j > k$.

The usual way of dealing with inequalities when we want to obtain a QUBO formulation is to add slack variables to transform them into equality constraints.

In the two cases of the transitivity constraints mentioned above, we have a sum of three terms that can take value 0 or 1 and must add up to at most 2. If we denote each of the three addends on the left-hand side of the constraint by x, y and

z, then we have a generic constraint of the form $x + y + z \leq 2$. We can introduce binary slack variables w and t for each such constraint so that we have a new and equivalent constraint

$$x + y + z + w + t = 2.$$

We can transform this constraint into a penalty term that we add to the objective function by considering

$$A(x + y + z + w + t - 2)^2,$$

where A is a large enough constant. On the one hand, if $x = y = z = 1$, then the value of the penalty term will be at least A for any choice of w and t. On the other hand, for any solution that verifies all the constraints, we can find w and t such that the value of the penalty term is 0.

The main problem with this approach is that we need to introduce two new variables for each transitivity constraint, of which we have $\frac{n(n-1)(n-2)}{3}$ (we first choose three different indices i, j, and k out of the n possibilities and then we create one constraint to avoid the cycle $i \to j \to k \to i$ and another to avoid the cycle $i \to k \to j \to i$). Although this is still a polynomial reduction of the original problem to a QUBO problem, the growth of the number of variables can easily become unfeasible for current quantum computers since we need one qubit for each variable.

In fact, the total number of variables that we need to consider with this formulation when there are n objects to be ranked is $\frac{n^2-n}{2} + \frac{2n(n-1)(n-2)}{3}$. For instance, when $n = 5$, this already involves 50 variables.

4.3 | Formulation with multiplicative constraints (MULT)

Consider the formulation introduced in the previous subsection and focus on one of the inequality constraints of the form $x + y + z \leq 2$. Since x, y, and z are binary, the constraint is equivalent to

$$xyz = 0.$$

Thus, we can consider the penalty term

$$Axyz,$$

which will be nonzero only in case $x = y = z = 1$.

The main advantage of this approach is that it does not introduce any new variable. It may seem that it can only be used directly with QAOA and not in quantum annealers since now we have terms of degree three in the objective function. However, there is an additional simplification that we can perform. Note that the constraints that we need in order to avoid cycles with the objects i, j, and k are

$$T_{ij}T_{jk}T_{ki} = T_{ij}T_{jk}(1 - T_{ik}) = 0$$

and

$$T_{ik}T_{kj}T_{ji} = T_{ik}(1 - T_{jk})(1 - T_{ij}) = 0.$$

Since the value on the left-hand side of these constraints is either 0 or 1, both constraints hold at the same time if and only if

$$T_{ij}T_{jk}(1 - T_{ik}) + T_{ik}(1 - T_{jk})(1 - T_{ij}) = 0,$$

which is a polynomial constraint of degree two because the terms of degree three cancel each other.

Thus, this alternative formulation, which is original to this work, only involves $\frac{n^2-n}{2}$ variables. Since we are using constraints that use multiplicative terms, we will call this formulation the MULT formulation. Of course, this formulation can be used for any problem whose solutions are order relationships, and not only for the ranking aggregation problem. For that kind of task, this formulation offers a reduction in the number of variables that need to be considered. As we will show in Section 5, this can mean a boost in the effectiveness of the method.

4.4 | Formulation via permutations (PERM)

Another way of representing a total order is by explicitly stating the position of each object $i \in \{1, \ldots, n\}$, that is, by representing a permutation of the objects 1 through n (see, e.g., earlier studies [40, 41]). To do this, we define binary variables p_{ij} for $1 \le i, j \le n$ such that $p_{ij} = 1$ if and only if object i appears at position j. This imposes the constraint that, for fixed i, exactly one variable p_{ij} equals 1 and for fixed j, exactly one variable p_{ij} equals 1. We can express these constraints as penalty terms in the QUBO framework as

$$A\left(\sum_{j=1}^{n} p_{ij} - 1\right)^2,$$

for every i, and

$$A\left(\sum_{i=1}^{n} p_{ij} - 1\right)^2,$$

for every j, where, in all cases, A is a large enough constant. The cost function in this case is

$$\sum_{i,j=1}^{n} c_{ij} \sum_{k=1}^{n-1} \sum_{l=k+1}^{n} p_{ik}p_{jl},$$

where c_{ij} is the cost of i being before j in the total order. Notice that this is, directly, a quadratic function and, hence, we have a QUBO problem that we can solve with either QAOA or quantum annealers.

The number of variables in this formulation is always n^2, but the number of terms in the Hamiltonian could be larger than in the approach described in the previous subsection. More concretely, the MULT formulation can only have quadratic terms of the forms $T_{ij}T_{jk}$, $T_{ij}T_{ik}$, and $T_{jk}T_{ik}$, where $i < j < k$. In total, it will have at most $3\binom{n}{3}$ quadratic terms. However, the PERM formulation can have quadratic terms of the forms $p_{ij}p_{ik}$ and $p_{ij}p_{kj}$ for any i, j, k. The number of such terms is $2n^3$, which is larger than $3\binom{n}{3}$.

An additional advantage of this formulation is that it may be used with recent modifications of the QAOA in which the condition of having a permutation represented by the binary variables p_{ij} is first established by an adequate preparation of the initial quantum state and then preserved in the application of all the subsequent operations of the quantum circuit (see earlier studies [41, 42]). However, such techniques involve changing the mixer Hamiltonian (denoted H_i in Section 3). Since that Hamiltonian is fixed on quantum annealers, it is not possible to use that approach on those devices. In this work, we focus only on formulations that can be used both with QAOA and quantum annealers.

4.5 | Formulation via domain wall encoding (WALL)

Domain wall encoding [43] is a technique that can be used to reduce the number of qubits needed in order to encode a series of binary variables of which one and only one is set to value 1 at a given moment. We encode the position of an object in a permutation of n objects by using $n + 1$ variables s_i ($i \in \{0, \dots, n\}$) taking values in $\{-1, 1\}$ subject to the constraint

$$-\sum_{i=0}^{n-1} s_i s_{i+1} = -(n-2),$$

with $s_0 = -1$ and $s_n = 1$. This forces the sequence $s_0, s_1, \dots, s_n$ to be monotone and to have just one sign change that we can use to encode the position of a given object.

In this way, we can use a formulation like the one proposed in the previous subsection but with a sequence of variables $s_0, \dots, s_n$ to encode the position of each object. Since $\sum_{i=0}^{n-1} s_i s_{i+1}$ is always at most $(n-2)$, we have that $-\sum_{i=0}^{n-1} s_i s_{i+1} + n - 2$ is always nonnegative and it is 0 only if the constraint is met. Thus, we can add the following term to the objective function

$$A\left(-\sum_{i=0}^{n-1} s_i s_{i+1} + n - 2\right),$$

where A is a large enough constant.

If the constraint is satisfied, we can recover the position of the object through

$$\sum_{i=0}^{n-1} \frac{s_{i+1} - s_i}{2},$$

so we can rewrite the column constraints and the objective function of the permutation formulation in terms of the variables s_i and map them to Z_i matrices to obtain an Ising formulation of the problem that we can use with either quantum annealing or QAOA.

With this approach, we reduce the number of variables of the permutation formulation from n^2 to $n^2 - n$. We will refer to this formulation as the WALL formulation.

5 | EXPERIMENTS AND RESULTS

In this section, we present some of the experiments that we have carried out with quantum simulators and actual quantum computers in order to compare the different formulations of the ranking aggregation problem presented above.

We have considered two different scenarios. The first one is the simplest one and only involves three objects and three cases of voting situations (unanimous voting, transitive tournaments, and nontransitive tournaments). Testing the first two cases allows us to evaluate how each of the different formulations behaves in cases in which the optimal solution is known beforehand. The second scenario involves cases with three, four, and five objects and cost matrices generated at random. This is a more general situation that can be used to evaluate the performance of the formulations on typical instances of the ranking aggregation problem.

The rest of this section is devoted to describing in detail the settings of each of both scenarios, to presenting the results that we have obtained, and to discussing their significance.

5.1 | First scenario

As a first way of testing the performance of the four problem formulations, we have considered three cases with three objects each, henceforth denoted as a, b, and c. The first case is a situation of unanimous voting, where we have 10 times the ranking $a > b > c$. In the second case, we have six times the ranking $a > b > c$ and four times the ranking $c > b > a$. In this case, all pairwise comparisons between objects are consistent with each other, avoiding the so-called Condorcet paradox and yielding a transitive tournament. Finally, for the third case, we have selected a situation with a cyclic tournament where we have four times the ranking $a > b > c$, 3 times the ranking $b > c > a$, and three times the ranking $c > a > b$. Notice that, in all these three cases, the optimal solution is the same: $a > b > c$. However, in each of them, the difficulty of computing the ranking is very different since in the first case there is a solution with cost zero, in the second case, the solution could be obtained by simply computing the associated tournament, while in the third case, (in which there is a cyclic tournament) obtaining the solution should be the most challenging, regardless of the chosen formulation.

For the experiments, we have computed the cost matrix for the unanimous voting, the transitive tournament, and the cyclic tournament, and we have used them to pose the ranking aggregation problem under the four formulations described in the previous sections.

An additional parameter that we need to select is the constant A that we use in the penalty terms of the QUBO formulations (see Section 4). The ratio of this penalty factor to the other coefficients in the cost function can affect the performance of quantum annealing (see Section "Scale of Biases" on page 21 of D-WAVE [44]). In order to test this effect, we have conducted our experiments with a range of values for A. In fact, we derive the values for A from the cost C of the solution in which the objects are ordered from 1 to n and then we compute $A = fC + 1$, where f takes values in $\{1, 2, 3, 10\}$. This is a way of deterministically setting the value of A in a manner that guarantees that any solution that does not satisfy one of the constraints will have higher cost than at least one of the feasible solutions (notice the inclusion of the "plus one" term).

For comparison, we also have included a case in the experiments in which we set A to the optimal value (that we compute by brute force) plus one (we denote this option by $f = 0$). Of course, this is something that cannot be done in practice when n is large, but we use it here as a useful benchmark.

We have used both quantum annealing and the QAOA algorithm to sample possible solutions to the resulting QUBO problems. In the case of quantum annealing, we have used a real quantum device accessible through D-Wave Leap [1].

[1]https://cloud.dwavesys.com/leap/

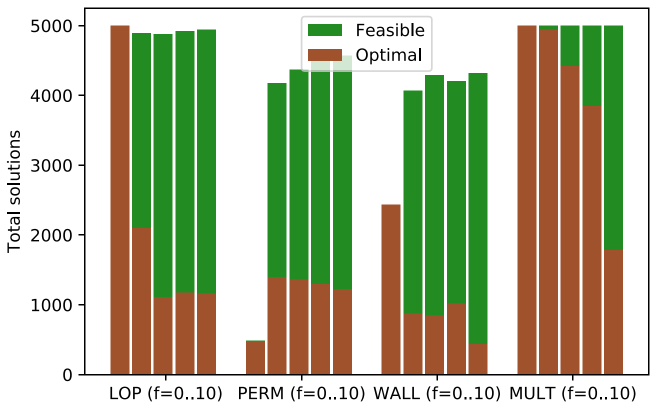

FIGURE 1 Quantum annealing results for the case of unanimous voting. [Colour figure can be viewed at wileyonlinelibrary.com]

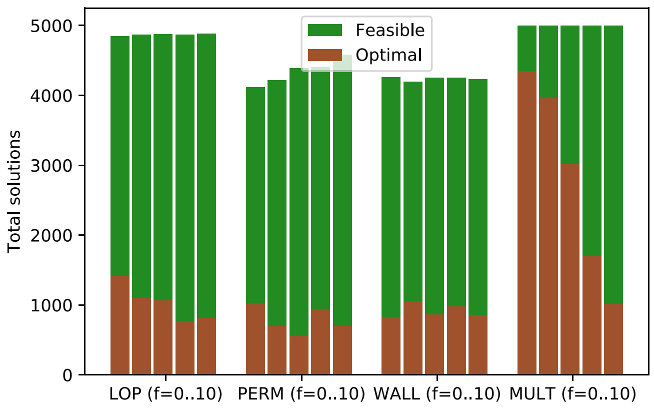

FIGURE 2 Quantum annealing results for the case of the transitive tournament. [Colour figure can be viewed at wileyonlinelibrary.com]

Namely, we have used the Advantage System 4.1, which has more than 5000 qubits and implements the Pegasus topology (more details about this quantum computer can be found at https://docs.dwavesys.com/docs/latest/doc_physical_properties.html). We used the default parameters and sampled 5000 solutions for each problem setting. For QAOA, we have used noiseless simulation with the Qiskit implementation of the algorithm [45]. We have used depth $p = 1$ and selected COBYLA [46] as optimizer (one of the options recommended in Fernández-Pendás et al. [38]). The rest of the parameters are the default ones. To factor in the variability inherent to the randomness of the algorithm (namely, the initial values of the circuit parameters and the stochastic decisions of the optimization algorithm), we repeated each experiment 100 times and computed the average probability of obtaining each solution, as well as the associated standard deviation. We also conducted experiments with $p \in \{2, 3\}$, but the trend of the results is completely analogous to that of the case with $p = 1$. For that reason, we only report here the results with $p = 1$.

In Figures 1–3, we present the results of the quantum annealing experiments. For each problem formulation (LOP, PERM, WALL and MULT), we show the total number of times that the optimal solution was found, together with the total number of times that a feasible solution was sampled. Notice, however, that the notion of what constitutes a feasible solution in the case of the LOP formulation is ambiguous. Since we need to introduce slack variables (cf. Section 4.2), it may be the case that a solution is unfeasible because those slack variables do not have the correct values to satisfy the equality constraints, but the original variables do satisfy the original constraints. In that situation, we have chosen to consider the solution as feasible (a similar decision is taken in Plewa et al. [28]) and compute the cost of the original variables, without taking into account the slack variables.

Also notice that, in each figure, we present five different bars for each of the formulations. They correspond to the different values of f that we have considered in order to compute the penalty constant A, as described above. These values are 0, 1, 2, 3, and 10 and are shown in ascending order from left to right.

As can be seen, for the simplest case (unanimous voting), both the LOP and the MULT formulations obtain perfect results when $f = 0$. However, for more realistic settings (with $f > 0$), the MULT formulation is clearly the best one: It consistently samples feasible solutions in each and every case, and it obtains the optimal solution with very high prob-

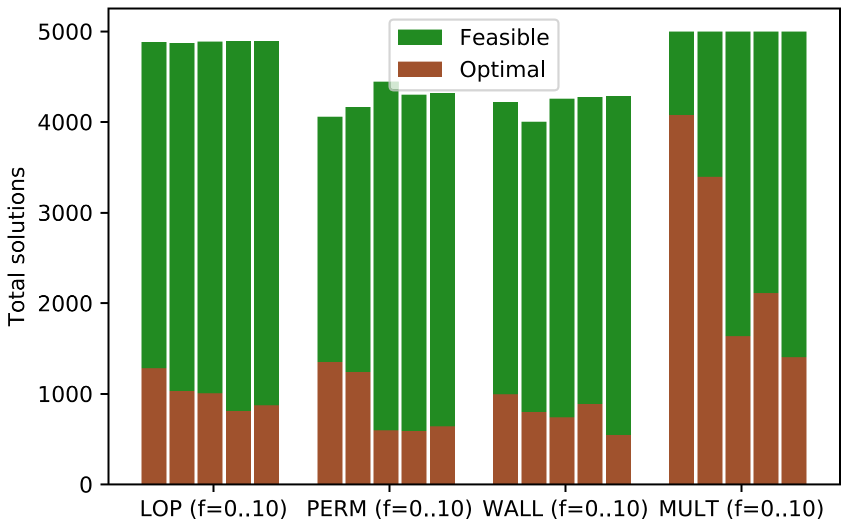

FIGURE 3 Quantum annealing results for the case of the cyclic tournament. [Colour figure can be viewed at wileyonlinelibrary.com]

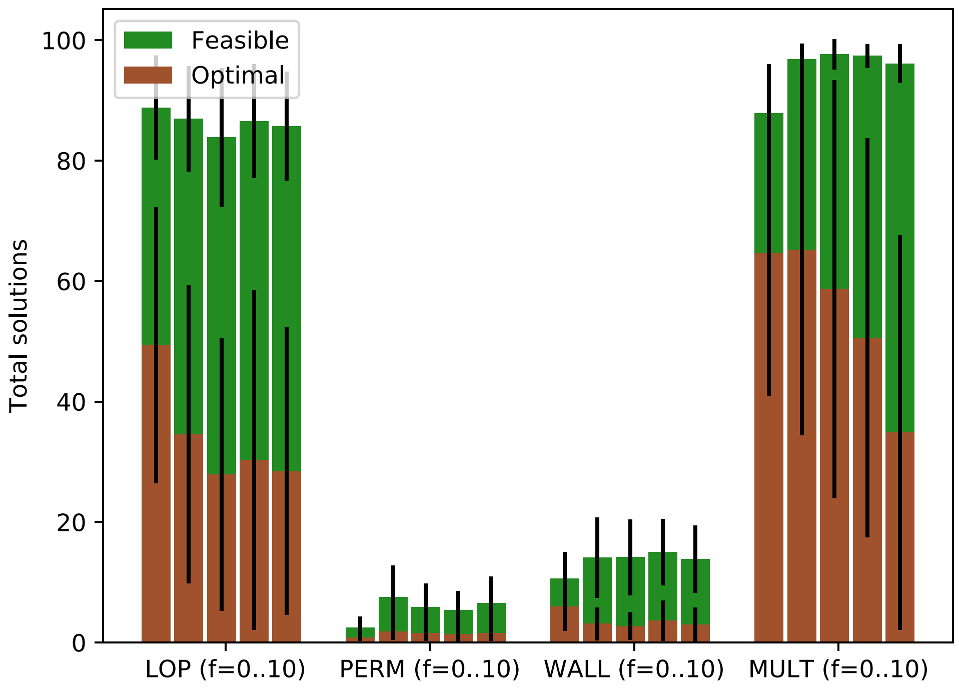

FIGURE 4 Quantum Approximate Optimization Algorithm (QAOA) results for the case of unanimous voting. [Colour figure can be viewed at wileyonlinelibrary.com]

ability. As expected, the performance decreases when f is set to larger values, so great care must be put in choosing the values of the penalty constant.

The PERM and WALL formulations behave much more poorly even on this simple problem. In fact, they fail to obtain feasible solutions in many cases, and their probability of obtaining the optimal solution is low compared with that of MULT. What is more, in the case $f = 0$, they find it difficult to obtain feasible solutions other than the optimal one. The reason for this is that, with these formulations, low penalty constants can lead to situations in which unfeasible solutions that are close to the optimal solution achieve low energy levels and are selected instead of other solutions that are feasible but have a higher cost.

In the two other cases (the transitive and cyclic tournaments), the performance of the MULT formulation is clearly superior to those of the other three formulations. MULT always samples feasible solutions, and the probability of obtaining

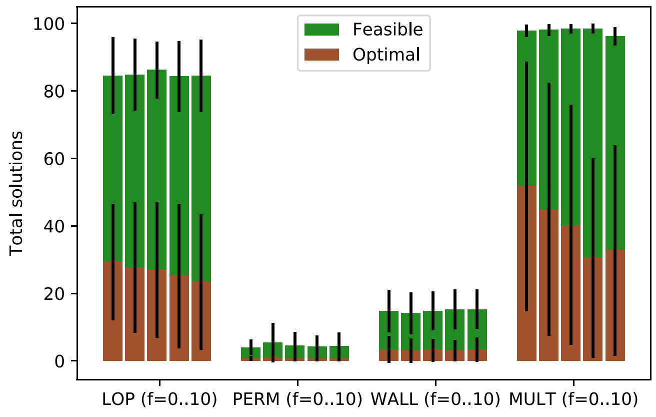

FIGURE 5 Quantum Approximate Optimization Algorithm (QAOA) results for the case of the transitive tournament. [Colour figure can be viewed at wileyonlinelibrary.com]

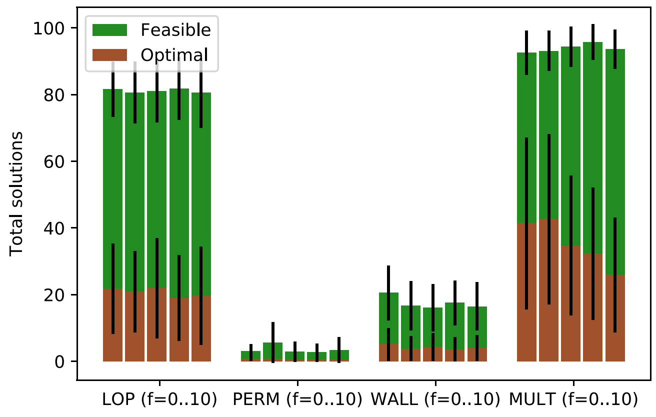

FIGURE 6 Quantum Approximate Optimization Algorithm (QAOA) results for the case of the cyclic tournament. [Colour figure can be viewed at wileyonlinelibrary.com]

the optimal solution is consistently higher than that of the other methods. We observe, once more, that the performance decreases with larger values of f. Despite that effect, MULT offers the best results for any value of f.

In Figures 4–6, we present the results for the QAOA experiments. We show the probability of obtaining a feasible solution together with the probability of obtaining the optimal solution. In this case, since we have conducted 100 experiments, with different initial conditions, for each problem, we also show the corresponding standard deviations. The same remarks that we made above apply here: For the LOP formulation, we consider as feasible those solutions that satisfy the constraints after removing slack variables, and we present bars from left to right that correspond to increasing values of f.

The results follow a trend that is similar to that observed in the quantum annealing experiments. MULT is, once again, the best performing formulation for the three problems, followed by LOP. PERM and WALL obtain very poor results. In general, the results are worse than with the quantum annealers. We have observed, for instance, that although LOP achieves a high probability of sampling a feasible solution, it selects the solution with the highest cost (the inverse of the optimal solution) about 20% of the times. These bad results in comparison with quantum annealing may be caused by using $p = 1$ in QAOA, which results in a very shallow quantum circuit.

In any case, MULT is, by far, the best formulation for the three problems. As with the quantum annealing experiments, the performance tends to degrade when increasing f. However, this effect is less noticeable than in the case of the quantum annealers.

5.2 | Second scenario

To test the performance of the different formulations in a more general setting, we have considered experiments in which we generate the values c_{ij} of the cost function by first generating at random partial votes and then aggregating them. To

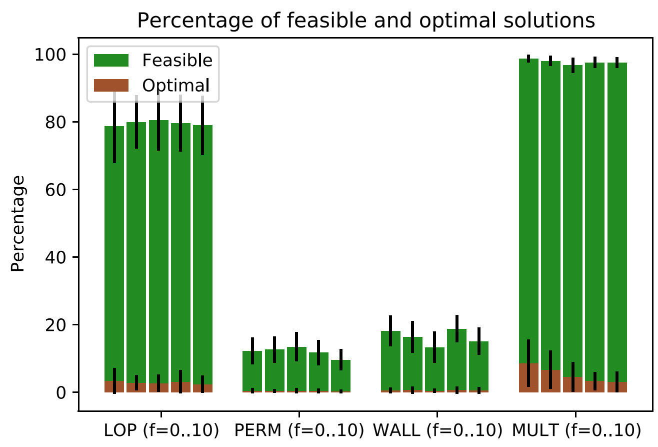

FIGURE 7 Mean value and standard deviation of the probability of obtaining feasible and optimal solutions in the case $n = 5$ with the quantum annealer and the four formulations for increasing values of f. [Colour figure can be viewed at wileyonlinelibrary.com]

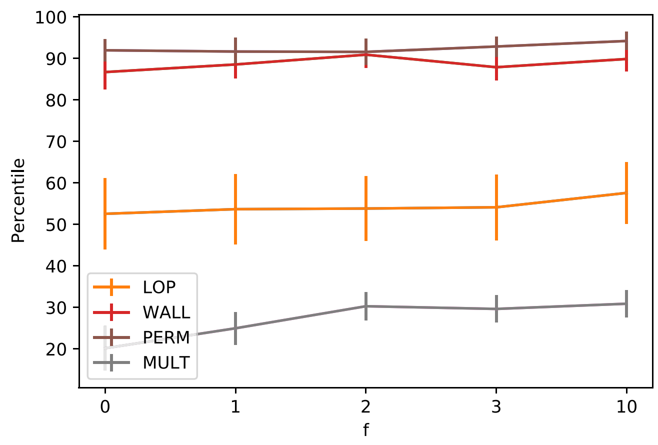

FIGURE 8 Mean value and standard deviation of the solution's percentile in the case $n = 5$ with the quantum annealer and the four formulations for increasing values of f. [Colour figure can be viewed at wileyonlinelibrary.com]

generate the partial votes, we generate binary matrices uniformly at random and reject those that do not represent partial orders (see [47] for more details on the topic).

With this method, we generated 50 different cost matrices with number of objects $n \in \{3, 4, 5\}$ and number of rankings $m = 10$. We used the four formulations of Section 4 to pose the ranking aggregation problem associated with the cost matrices and next we solved them with both a quantum annealer and with the QAOA algorithm. The settings are exactly the same ones as in the experiments reported in the previous subsection.

In Figures 7 and 8, we present data on the results of the experiments with the quantum annealer for the case $n = 5$ (the cases $n = 3$ and $n = 4$ follow exactly the same trends, so we restrict ourselves to reporting our findings in the problems with the largest number of objects, which is the hardest to solve).

Figure 7 is similar to the ones presented in the previous subsection. It shows the mean value and standard deviation, over the 50 cost matrices, of the probability of obtaining the optimal solution and of sampling a feasible solution. Again, the bars, from left to right, correspond to increasing values of f. As we can see, MULT clearly outperforms the other formulations in both obtaining feasible and optimal solutions. We also observe that the performance decreases for high values of f, highlighting once again the importance of choosing its value with care.

In Figure 8, we present the average value and standard deviation of the cost percentile of the solutions obtained with the four formulations for increasing values of f. To determine the percentile of a solution, we have computed all the feasible solutions and ordered them by increasing cost. In those cases in which one of the samples represented an unfeasible solution, we considered its associated percentile to be 100. As we can see, MULT consistently obtains the lowest percentiles, corresponding to solutions with overall lowest cost.

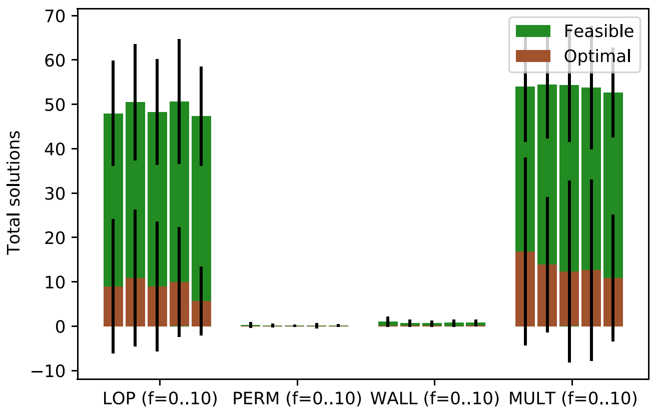

FIGURE 9 Mean value and standard deviation of the probability of obtaining feasible and optimal solutions in the case $n = 4$ with Quantum Approximate Optimization Algorithm (QAOA) and the four formulations for increasing values of f. [Colour figure can be viewed at wileyonlinelibrary.com]

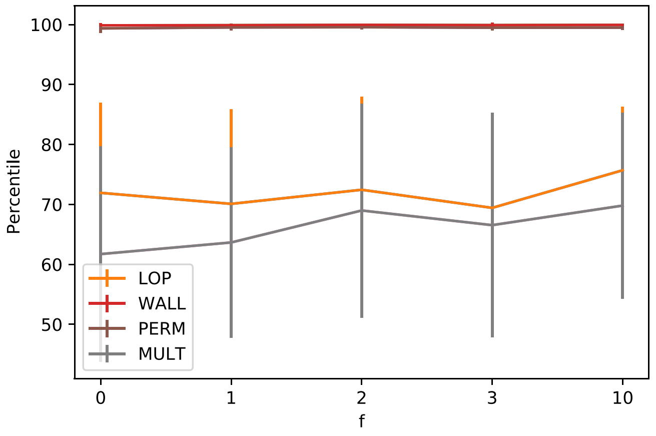

FIGURE 10 Mean value and standard deviation of the solution's percentile in the case $n = 4$ with Quantum Approximate Optimization Algorithm (QAOA) and the four formulations for increasing values of f. [Colour figure can be viewed at wileyonlinelibrary.com]

Figures 9 and 10 show the results for the application of QAOA with the four formulations and the case $n = 4$. The case with $n = 5$ could not be run, because it required 50 qubits for the LOP formulation. Simulating this number of qubits would involve manipulating vectors of size 2^{50}, which might be out of reach even for the world biggest supercomputers (the famous quantum supremacy experiments conducted by Google researchers were done with a 53-qubit quantum computer [22]).

As can be seen on both Figures 9 and 10, the performance of the PERM and WALL formulations is abysmal. They rarely find feasible solutions, let alone the optimal one. MULT is, again, the best of the four formulations. LOP is a little bit behind, but we also need to take into account that the number of qubits required for this formulation scales much worse than that of MULT. In fact, as we have mentioned, for $n = 5$, it is almost impossible to simulate the type of circuit required by LOP, while we can still do it for MULT on a regular laptop.

6 | CONCLUSIONS

In this paper, we have introduced four different formulations of the Kemeny ranking aggregation problem that can be used with quantum algorithms in order to solve it. To the best of our knowledge, this is the first time that this problem is approached with quantum techniques.

Two of the formulations that we have developed are based on finding the binary matrix that represents the total order that we consider as a candidate solution to the ranking problem. One of them, that we call LOP, is based on transforming a binary linear programming problem into a QUBO problem. The other, called MULT, substitutes some linear constraints with polynomial ones that are eventually reduced to quadratic constraints. The other two formulations are based on representing the candidate solutions as permutations of the objects. In the PERM formulation, we represent these permutations by means of binary variables that indicate the positions of the objects in the ranking. For the WALL permutation, we consider a representation of these variables through the domain-wall encoding technique.

To evaluate the effectiveness of these formulations, we have tested them under different scenarios both with quantum simulators and with actual quantum computers, using quantum annealing and the quantum approximate optimization algorithm (QAOA). The results of our experiments clearly show that the MULT representation outperforms all the rest on all the metrics that we have considered, including the number of variables, the probability of obtaining a feasible solution, the probability of obtaining an optimal solution, and the overall percentile of the solution's cost. This shows that in order to achieve the best possible results with quantum optimization algorithms, selecting a good formulation is crucial. In fact, for the ranking aggregation problem, selecting the text-book formulation via binary linear programs leads to representations with a number of variables that grows much more quickly than in case of the representation (MULT) that we have found to offer the best results. What is more, the technique used to develop the MULT formulation also works for other kinds of problems in which the solutions are order relationships.

In future work, we plan to explore the application of similar techniques to other related problems such as the aggregation of other types of binary relations [18, 19]. Note that this extension is not trivial since the consideration of different properties for a binary relation may lead to different constraints in the associated zero-one linear programming problem. For instance, the MULT representation presented in this paper needed to be carefully adapted to deal with the transitivity constraint. We are, as well, interested in studying whether the performance of the permutation-based representations can be improved by using modified versions of QAOA that have been recently introduced to deal with that kind of objects.

ACKNOWLEDGEMENTS

This work was supported in part under grant PID2020-119082RB-C22 funded by MCIN/AEI/10.13039/501100011033, grant AYUD/2021/50994 funded by Gobierno del Principado de Asturias, grant TIN2017-87600-P funded by Spanish MINECO, QUANTUM SPAIN project funded by the Ministry of Economic Affairs and Digital Transformation of the Spanish Government and the European Union through the Recovery, Transformation and Resilience Plan - NextGenerationEU and from the Flemish Government under the “Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen” program.

CONFLICT OF INTEREST STATEMENT

This work does not have any conflicts of interest.

ORCID

José Ranilla https://orcid.org/0000-0003-2941-3741

REFERENCES

1. K. J. Arrow, A. K. Sen, and K. Suzumura, *Handbook of social choice and welfare*, Vol. **1**, North-Holland, Amsterdam, 2002.
2. A. Sen, *Social choice theory*, Handb. Math. Econ. **3** (1986), 1073–1181.
3. S. Lin, *Rank aggregation methods*, Wiley Interdiscip. Rev. Comput. Stat. **2** (2010), no. 5, 555–570.
4. C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, *Rank aggregation methods for the web*, Proceedings of the 10th International Conference on World Wide Web, Hong Kong, 2001, pp. 613–622.
5. K. J. Arrow, *Social choice and individual values*, Second, Yale University Press, New Haven, 1963.
6. J. G. Kemeny, *Mathematics without numbers*, Daedalus **88** (1959), no. 4, 577–591.
7. H. P. Young, *Condorcet's theory of voting*, Am. Polit. Sci. Rev. **82** (1988), no. 4, 1231–1244.
8. H. P. Young and A. Levenglick, *A consistent extension of Condorcet's election principle*, SIAM J. Appl. Math. **35** (1978), no. 2, 285–300.
9. J. Bartholdi, C. A. Tovey, and M. A. Trick, *Voting schemes for which it can be difficult to tell who won the election*, Soc. Choice Welf. **6** (1989), no. 2, 157–165.
10. A. Davenport and J. Kalagnanam, *A computational study of the Kemeny rule for preference aggregation*, Proceedings of the 19th national conference on artificial intelligence (AAAI04) vol. 4, 2004, pp. 697–702.
11. V. Conitzer, A. Davenport, and J. Kalagnanam, *Improved bounds for computing Kemeny rankings*. AAAI vol. 6, 2006, pp. 620–626.
12. I. Azzini and G. Munda, *A new approach for identifying the Kemeny median ranking*, Eur. J. Oper. Res. **281** (2020), no. 2, 388–401.
13. F. Schalekamp and A. Zuylen, *Rank aggregation: together we're strong*, 2009 Proceedings of the Eleventh Workshop on Algorithm Engineering and Experiments (ALENEX). SIAM, 2009, pp. 38–51.

14. M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information: 10th anniversary edition*, Cambridge University Press, 2011.
15. P. Emerson, *The original Borda count and partial voting*, Social Choice and Welfare **40** (2013), no. 2, 353–358.
16. D. Black, *Partial justification of the Borda count*, Public Choice **28** (1976), no. 1, 1–15.
17. R. Fagin, R. Kumar, and D. Sivakumar, *Comparing top k lists*, SIAM J. Discret. Math. **17** (2003), no. 1, 134–160.
18. J. P. Barthelemy and B. Monjardet, *The median procedure in cluster analysis and social choice theory*, Math. Soc. Sci. **1** (1981), no. 3, 235–267.
19. J. Drewniak and U. Dudziak, *Preservation of properties of fuzzy relations during aggregation processes*, Kybernetika **43** (2007), no. 2, 115–132.
20. C. C. McGeoch, *Adiabatic quantum computation and quantum annealing*, Morgan&Claypool Publishers: Synthesis Lectures on Quantum Computing, 2014.
21. E. Farhi, J. Goldstone, and S. Gutmann, *A quantum approximate optimization algorithm*, 2014. arXiv preprint arXiv:1411.4028.
22. F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, and B. Burkett, *Quantum supremacy using a programmable superconducting processor*, Nature **574** (2019), no. 7779, 505–510.
23. O. Salehi, A. Glos, and J. A. Miszczak, *Unconstrained binary models of the travelling salesman problem variants for quantum optimization*, Quantum Inf. Process **21** (2022), no. 2, 1–30.
24. S. Gonzalez-Bermejo, G. Alonso-Linaje, and P. Atchade-Adelomou, *GPS: a new TSP formulation for its generalizations type QUBO*, Mathematics **10** (2022), no. 3, 416.
25. C. Papalitsas, T. Andronikos, K. Giannakis, G. Theocharopoulou, and S. Fanarioti, *A QUBO model for the traveling salesman problem with time windows*, Algorithms **12** (2019), no. 11, 224.
26. Z. Tabi, K. H. El-Safty, Z. Kallus, P. Hága, T. Kozsik, A. Glos, and Z. Zimborás, *Quantum optimization for the graph coloring problem with space-efficient embedding*, 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), 2020, pp. 56–62.
27. R. Quintero, D. Bernal, T. Terlaky, and L. F. Zuluaga, *Characterization of QUBO reformulations for the maximum k-colorable subgraph problem*, Quantum Inf. Process **21** (2022), no. 3, 1–36.
28. J. Plewa, J. Sieńko, and K. Rycerz, *Variational algorithms for workflow scheduling problem in gate-based quantum devices*, Comput. Inform. **40** (2021), no. 4, 897–929.
29. A. I. Pakhomchik, S. Yudin, M. R. Perelshtein, A. Alekseyenko, and S. Yarkoni, *Solving workflow scheduling problems with QUBO modeling*, 2022. arXiv preprint arXiv:2205.04844.
30. H. Irie, G. Wongpaisarnsin, M. Terabe, A. Miki, and S. Taguchi, *Quantum annealing of vehicle routing problem with time, state and capacity*, International Workshop on Quantum Technology and Optimization Problems, 2019, pp. 145–156.
31. S. Harwood, C. Gambella, D. Trenev, A. Simonetto, D. Bernal, and D. Greenberg, *Formulating and solving routing problems on quantum computers*, IEEE Trans. Quantum Eng. **2** (2021), 1–17.
32. R. Martí and G. Reinelt, *The linear ordering problem: exact and heuristic methods in combinatorial optimization*, Applied Mathematical Sciences, Vol. **175**, Springer, 2011.
33. K. O. May, *A set of independent necessary and sufficient conditions for simple majority decision*, Econometrica: J. Econ. Soc. **20** (1952), 680–684.
34. B. Monjardet, *An axiomatic theory of tournament aggregation*, Math. Oper. Res. **3** (1978), no. 4, 334–351.
35. A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'bien, *A variational eigenvalue solver on a photonic quantum processor*, Nat. Commun **5** (2014), no. 1, 1–7.
36. E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, *A quantum adiabatic evolution algorithm applied to random instances of an NP-Complete problem*, Science **292** (2001), no. 5516, 472–475.
37. M. Born and V. Fock, *Beweis des adiabatensatzes*, Z. Phys. **51 (3-4)** (1928), 165–180.
38. M. Fernández-Pendás, E. F. Combarro, S. Vallecorsa, J. Ranilla, and I. F. Rúa, *A study of the performance of classical minimizers in the quantum approximate optimization algorithm*, J. Comput. Appl. Math. **404** (2022), 113388.
39. E. F. Combarro and S. González-Castillo, *A pratical guide to quantum machine learning and quantum optimization*, Packt, Birminghan, 2023.
40. A. Lucas, *Ising formulations of many np problems*, Frontiers Phys. **2** (2014), 5.
41. S. Hadfield, Z. Wang, B. O'gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, *From the quantum approximate optimization algorithm to a quantum alternating operator ansatz*, Algorithms **12** (2019), no. 2, 34.
42. A. Bärtschi and S. Eidenbenz, *Grover mixers for QAOA: shifting complexity from mixer design to state preparation*, 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), 2020, pp. 72–82.
43. N. Chancellor, *Domain wall encoding of discrete variables for quantum annealing and QAOA*, Quantum Sci. Technol. **4** (2019), no. 4, 45004.
44. D-WAVE, *D-wave problem-solving handbook*, 2021.
45. M. S. Anis, H. Abraham, R. A. AduOffei, G. Agliardi, M. Aharoni, I. Y. Akhalwaya, G. Aleksandrowicz, T. Alexander, M. Amy, and S. Anagolum, *Qiskit: an open-source framework for quantum computing*, 2021.
46. M. J. D. Powell, *A direct search optimization method that models the objective and constraint functions by linear interpolation*, Advances in optimization and numerical analysis, 1994, pp. 51–67.
47. P. C. Fishburn, *Induced binary probabilities and the linear ordering polytope: a status report*, Math. Soc. Sci. **23** (1992), no. 1, 67–80.

How to cite this article: E. F. Combarro, R. Pérez-Fernández, J. Ranilla, and B. De Baets, *Solving the Kemeny ranking aggregation problem with quantum optimization algorithms*, Math. Meth. Appl. Sci. **46** (2023), 17065–17081, DOI 10.1002/mma.9489.