scientific reports

OPEN

Modeling routing problems in QUBO with application to ride-hailing

Michele Cattelan[1,2]✉ & Sheir Yarkoni[1]

Many emerging commercial services are based on the sharing or pooling of resources for common use with the aim of reducing costs. Businesses such as delivery-, mobility-, or transport-as-a-service have become standard in many parts of the world, fulfilling on-demand requests for customers in live settings. However, it is known that many of these problems are NP-hard, and therefore both modeling and solving them accurately is a challenge. Here we focus on one such routing problem, the Ride Pooling Problem (RPP), where multiple customers can request on-demand pickups and drop-offs from shared vehicles within a fleet. The combinatorial optimization task is to optimally pool customer requests using the limited set of vehicles, akin to a small-scale flexible bus route. In this work, we propose a quadratic unconstrained binary optimization (QUBO) program and introduce efficient formulation methods for the RPP to be solved using metaheuristics, and specifically emerging quantum optimization algorithms.

Keywords Quantum computing, Combinatorial optimization, Unconstrained binary optimization

Solving combinatorial optimization problems by mapping them to quadratic unconstrained binary optimization (QUBO) problems is a well-studied field in literature[1,2]. The topic has gained increased attention recently in light of advances in quantum optimization, and consequentially, other quantum- and physics-inspired metaheuristic optimization algorithms[3–5]. Furthermore, rapid development in the field of quantum technologies has produced prototype quantum processors for public use which are able to execute these quantum algorithms, making them interesting tools to test the power of quantum computing. Companies such as Amazon, Google, D-Wave Systems, IBM, among others, all have cloud-accessible processors that have been examined by researchers and practitioners alike from all around the world[6–8]. The promise of quantum computing lies in the theoretical demonstration of some quantum algorithms being able to outperform their classical counterparts for particular classes of problems[7,9]. Instead of classical bits in binary programming which can only be 0 or 1 deterministically, quantum computers (QCs) are built from *quantum bits*, or qubits, which have probabilities of being measured as 0 and 1. These qubits allow for the construction of new classes of algorithms where quantum mechanics are exploited to perform tasks that are fundamentally inefficient for classical computers to perform[10]. Specifically for optimization, quantum computing researchers have started examining well-known hard optimization problems in hopes of finding quantum algorithms that can utilize the potential of QCs to gain performance advantages relative to classical optimization algorithms. Currently, the most promising such algorithms are quantum annealing (QA)[5] and the quantum approximate optimization algorithm (QAOA)[4], two metaheuristic quantum optimization algorithms motivated by the adiabatic theorem[3]. Both of these algorithms have been implemented in quantum hardware and tested in academic and industrial settings over a variety of optimization problems[11–14]. Solving optimization problems using these algorithms requires problems to be formulated either as QUBOs or Ising models, which are isomorphic and NP-hard to minimize[15,16]. Furthermore, many canonical NP-hard and NP-complete problems have straightforward transformations to QUBO or Ising[2], making them of practical use for application development. However, the field of quantum computing hardware and quantum optimization algorithms is a rapidly-evolving field, the technical aspects of which are beyond the scope of this paper. For more information about quantum hardware and quantum optimization, we refer the reader to Ref.[8]. The limiting factor in formulating optimization problems using state-of-the-art quantum hardware is the number of qubits and the degree of connectivity between them. Therefore, optimization problems with high connectivity between variables require more layers in quantum circuits (meaning, deeper circuits), or an increase in the number of physical qubits used to represent problem variables, both of which are scarce resources. So, finding minimal QUBO representations that most efficiently use the limited resources can be of significant practical use.

[1]Volkswagen Data:Lab, Volkswagen AG, Munich 80805, Germany. [2]Institute for Theoretical Physics, University of Innsbruck, Innsbruck A-6020, Austria. ✉email: Michele.Cattelan@student.uibk.ac.at

In this paper, we focus on a particular class of NP-hard combinatorial optimization problems, routing problems, which are of practical relevance to modeling many real-world problems, from logistics and transportation to drilling holes in circuit boards, to planning factory floors. While the theoretical motivations for the Traveling Salesperson Problem (TSP), and by extension the Vehicle Routing Problem (VRP), are well-known, modeling complex routing problems often requires application-specific constraints to ensure that solutions are practically feasible. For example, constraints such as service time windows, vehicle capacities, or real-time traffic conditions all need to be modeled correctly such that the minimum of the QUBO corresponds to the global optima of the original problem. In our work, we investigate one such real-world routing problem, namely the Ride Pooling Problem (RPP), which is formally described as follows: given a fleet of vehicles A and a set of requests (s, f) with *pickup points* s and *drop-off points* f for n customers, assign routes to vehicles in A to pick up and drop off all customers such that the cost of the target (objective) function is minimized. Historically, the RPP had been used to solve flexible fleet-services problems, such as fuel delivery scheduling and optimization[17]. However, the investigation of RPP has gained renewed interest in recent years through the emergence of car-sharing services such as Uber and Lyft and is therefore of practical interest to both model and solve accurately in practice. As such, objective functions incorporating minimizing lateness of arrivals, waiting time, and similar constraints, are used when solving the RPP. In this work, we focus on solving the RPP using distance minimization of vehicles in the fleet to allow a straightforward motivation (and comparison) with TSP and VRP QUBO models. The basic methods developed in this paper can extend to almost any of these objective functions, and therefore are general enough to be used for a wide variety of applications. Specifically, we show how to construct QUBOs that accurately represent the RPP problem through the inclusion of problem-specific constraints, compare this to canonical representations of TSP and VRP variants in QUBO, and estimate the resources required to solve such RPP QUBOs using quantum processors and quantum algorithms. We demonstrate how the methods we develop could lead to more resource efficient problem representations for quantum optimiziation algorithms.

The paper is organized as follows. In Sect. "Previous works", we review previous papers about QUBO formulations of routing problems primarily motivated by quantum algorithms. Following this, in Sect. "Routing problems in QUBO" we present an in-depth analysis of QUBO formulations for routing problems, and show that some choices of binary/decision variables yield impractical QUBO models for metaheuristic optimization algorithms. Finally, in Sect. "The ride-pooling problem QUBO" we motivate the constraints required to accurately model the RPP, and present our QUBO formulation of the problem.

Previous works

We briefly present previous works where TSPs/VRPs (and similar problems) were modeled as QUBOs specifically for quantum optimization (or similarly inspired) algorithms. In Ref.[2], straightforward representations of Hamiltonian paths, cycles, and therefore TSPs were introduced for binary optimization with either quantum or similar such metaheuristic optimization algorithms. The set of binary decision variables used for all these encode whether each location was visited at a particular step in the path, thus requiring $O(N^2)$ variables for an n-location TSP problem (or n node graph, for Hamiltonian path/cycle problems in general). Feld et al.[18] extended this work to the Capacitated VRP (CVRP), showing how to construct a VRP QUBO with capacities. The TSP can be easily extended to VRP using $O(kN^2)$ (where k is the number of vehicles) QUBO variables using the same decision variables as in Ref.[2]. To include the capacity constraints, the authors transform inequalities to equalities with slack variables for QUBO using terms from the knapsack QUBO formulation in Ref.[2]. The authors explore the trade-offs between modelling parts of the problem with QUBOs independently versus solving the CVRP as a single combined QUBO.

Similarly, additional work from Ref.[19] in this direction further showed how to construct CVRP QUBOs from known benchmark data. Several QUBOs were used to partition the problem starting from the properties of the data. The proposed methods were developed starting from the classical optimization literature. They proposed a clustering algorithm to shard the set of locations of the VRP and reduce the problem to a set of TSPs, then modelled as QUBOs. The solutions of these TSP QUBOs can be combined into solutions to the original VRP problem. Furthermore, they developed a method to cluster the locations so that each cluster's capacity constraints can be satisfied. These QUBOs were benchmarked against the original QUBO formulation as presented in Ref.[2] as well as a new QUBO obtained by constraining the vehicles to visit an equal number of locations.

Another constraint, the *time-window*, was developed for QUBO routing problems in Ref.[20]. The problem to be solved consists of a TSP where some locations must be visited more than once. The authors introduce the concept of time windows to implement the fact that specific locations in the TSP problem need to be visited within certain times so that a solution is feasible. This implementation requires the decision variables to include both the information that an edge is visited and the information on the time this edge is visited. This was done similarly to the VRP QUBO implementation, where two indices were used to encode the edge and one index was used to encode the time frame. A similar work was proposed also by Ref.[21] to implement the QUBO model of constrained VRPs. In this case, the concept of time windows is used to implement several constraints that depend on the flow of time, e.g. the capacity constraint since the vehicles can carry different customers or goods during their route. The authors proposed to use a continuous time frame that is discretized to be implemented by decision variables. This formulation is helpful when implementing constraints with QUBOs because it avoids slack variables. However, the number of variables used depends on the discretization of time and it can become disadvantageous when the number of time windows needed to implement the problem becomes large.

Routing problems in QUBO

Quadratic unconstrained binary optimization (QUBO) is defined as follows: given a quadratic cost matrix Q ($N \times N$ real-values), find the minimum $0-1$ assignment to binary vector x (of length N) that minimizes the QUBO cost function. Formally, we say that the objective of the QUBO is:

$$\min_x x^T Qx. \tag{1}$$

The QUBO problem presented here is NP-hard to minimize in the worst case[16], and furthermore, QUBO is isomorphic to the NP-hard Ising model[2] under a change of variable. Ising models are a well-studied model from physics, and use spin vectors $s = \{-1, 1\}^N$ to formulate the objective function (represented by a so-called Hamiltonian H):

$$H(s) = \sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j. \tag{2}$$

Here, $s_i \in s$ are the spin variables, h_i are the associated linear cost terms and J_{ij} are the quadratic interaction terms. Thus, the task is to find a set of spin values such that the cost function (in physics known as the energy of the system) is minimized. Note that in the QUBO case, the linear terms are the diagonals of the matrix Q, as $x_i^2 = x_i, \forall x_i \in \{0,1\}$. In literature, the Ising model and QUBO are often used interchangeably depending on the problem being formulated. However, QUBOs are sometimes preferred in quantum optimization due to the simplicity of representing certain constraints. For example, the one-hot constraint over a set of binary variables $x_i \in x$ in QUBO is formulated as:

$$\sum_i x_i = 1 \iff \min \left(1 - \sum_i x_i\right)^2 = 0. \tag{3}$$

The right-hand side of the implication is then added to the objective function (with a high penalty) to ensure that the global minimum of the QUBO is reached when exactly one binary variable $x_i = 1$. Graph problems, routing problems, and similar optimization problems all require such constraints, and therefore are usually formulated as QUBOs rather than Ising models. For the remainder of our work, we operate strictly on binary variables to build our optimization problems using QUBOs, although the techniques and results presented hold equally for Ising models using the binary-to-spin variable change of basis.

As mentioned, routing problems are of particular interest for industrial optimization problems due to their wide applicability and have been studied extensively in literature. To motivate our work, we start by reviewing known QUBO formulations of routing problems. The Traveling Salesperson Problem (TSP) is a well-studied NP-hard combinatorial optimization problem with known approximation results[22]. Formally, TSP is defined as follows: given a weighted graph $G = (V, E)$ with $n = \text{card(V)}$ nodes (representing locations), find a cycle such that every node is visited exactly once with the minimum sum of weights of edges used (distances and connections between locations). This is also known as a (minimum weight) Hamiltonian cycle. To create the QUBO for TSP, we present the formulation described in Ref.[2]. The binary variables we use represent whether the location $v \in V$ is visited as the i-th location in the cycle:

$$x_{v,i} = \begin{cases} 1 & \text{if location } v \text{ is the } i\text{-th location to visit,} \\ 0 & \text{otherwise.} \end{cases}$$

Note that, in order to represent a cycle, we assume all sums over indices are taken to be modulo n. We now use these binary variables to define the objective function and constraints of the TSP QUBO. The objective function of the TSP is the sum of weights along the edges used in the graph; this is typically referred to as the distance. Letting $w_{v,k}$ be the weight of the edge (v, k) in G, the distance minimization can be represented as:

$$H_A = \sum_{(v,k)\in E} \sum_{i=0}^{n-1} \frac{w_{v,k}}{W} x_{v,i}\, x_{k,i+1}, \tag{4}$$

where $W := \epsilon + \max_{(v,k)\in E} w_{v,k}$ (where $\epsilon > 0$) is a normalization factor to bound the weights strictly between 0 and 1. We now continue to formulate the problem as a QUBO by enumerating the constraints. As explained above, in order to define a QUBO objective to minimize, the implementation of the constraints has to be included in the objective function directly. Notice that H_A alone is not sufficient, as we can easily see by inspection that the optimal solution is the zero vector with minimum value 0, since we are not constraining the values that $x_{v,i}$ can take. Therefore, we add the following constraints based on our binary variable definition: only one location can be visited at each step in the tour, and no location is visited twice during a tour. These two conditions result in two separate sets of one-hot constraints, which can be re-written for QUBO as follows:

$$H_B = \sum_{v\in V} \left(1 - \sum_{i=1}^{n} x_{v,i}\right)^2 + \sum_{i=1}^{n} \left(1 - \sum_{v\in V} x_{v,i}\right)^2. \tag{5}$$

The two summations in Eq. (5) are minimized when the two described constraints are valid. Thus, the global minima of the function given by the sum of Eqs. (4) and (5) are the tours that minimize both the distance function and satisfy the constraints defined above. Lastly, we implement the constraint that a tour of the locations is a valid Hamiltonian cycle of the graph. This happens only if the cycle is a subgraph of $G(V, E)$, meaning, we must forbid steps in the tour between nodes not connected in G.

In order to include this in the QUBO, we add a penalty term between all decision variables representing non-adjacent nodes in G:

$$H_C = \sum_{(v,w)\notin E} \sum_{i=1}^{n} x_{v,i}\, x_{w,i+1}. \tag{6}$$

From the normalization in Eq. (4), quadratic terms using QUBO variables representing adjacent nodes in G have lower cost than non-adjacent nodes, and therefore the condition of being a Hamiltonian cycle of the original graph is fulfilled. By summing all the contributions in Eqs. (4), (5) and (6) we obtain the QUBO formulation for TSP:

$$H_{TSP} = H_A + H_B + H_C. \tag{7}$$

A well-known generalization of the TSP is the Vehicle Routing Problem (VRP). This problem, which is also NP-hard and widely studied in literature, describes the case where a fleet of vehicles (starting from a common *depot*) must each construct a cycle such that a set of locations is only visited once and each vehicle returns to the depot at the end, subject to some minimization function. Therefore, TSP is a special case of VRP where the number of vehicles is one.

We can formulate VRP as a QUBO by reusing many of the parts introduced for TSP– we do this by considering a separate TSP sub-problem for each vehicle and apply global constraints to ensure the validity of the solution. Again, consider the graph $G = (V, E)$, where nodes are the set of all the locations (plus the depot), and weighted edges are distances between locations. For generality, we consider the case where G is complete, meaning every location can be reached from any other location. Thus, the binary decision variables are:

$$x_{a,v,s} = \begin{cases} 1 & \text{the vehicle } a \text{ is in location } v \text{ at step } s, \\ 0 & \text{otherwise.} \end{cases}$$

The index a enumerates the vehicles in the fleet, where A is the total number of vehicles. The indices s and i play the same role as in Eq. (7), where again we consider the sum over location indices modulo n (without loss of generality we define the first location $s = 0$ to be the depot).

Now, the objective function for the VRP QUBO is:

$$H_A^{\mathrm{VRP}} = \sum_{a=1}^{A} \sum_{(v,k)\in E} \sum_{s=0}^{n} \frac{w_{v,k}}{W} x_{a,v,s}\, x_{a,k,s+1}, \tag{8}$$

with W as defined for TSP. As we can easily see, if we consider the case where $A = 1$, we recover the objective function described in Eq. (4) for TSP. For the rest of the constraints, i.e. the generalizations of eq. (5), we have:

$$H_B^{\mathrm{VRP}} = \sum_{v\in V\setminus\{d\}} \left(1 - \sum_{a=1}^{A}\sum_{s=0}^{n} x_{a,v,s}\right)^2 + \sum_{a=1}^{A}\sum_{s=0}^{n}\left(1 - \sum_{v\in V} x_{a,v,s}\right)^2. \tag{9}$$

The one-hot constraints are the same as before, but with the addition (in the first summation) that the constraint sums over all vehicles; meaning, that every location can only be visited exactly once, shared between all vehicles. The only location excluded from this constraint is the depot– since we don't know *a priori* how long the optimal tours of each vehicle are (only that it is upper bounded by the number of locations), we must allow them to "stay" in the depot as long as they need to. The second part of Eq. (5) ensures that for each vehicle (the outer sum), only one location is visited at a time, as for TSP. For the final QUBO, the VRP QUBO is obtained by summing together Eq. (8) and eq. (9):

$$H^{\mathrm{VRP}} = H_A^{\mathrm{VRP}} + H_B^{\mathrm{VRP}}, \tag{10}$$

Again note that if we restrict the number of vehicles in the problem to one, then we recover the full TSP QUBO, resulting in correct QUBO representations for both VRP and TSP.

Alternative formulations of routing problems in QUBO

In the previous section, we briefly reviewed the representation of VRP and TSP in QUBO. However, it is important to note that encoding valid constraints in QUBO is inherently dependent on the choice of decision variables. In previous literature, two such possibilities have been described, notably the *node-based* decision variables and the *edge-based* ones. In the node-based version for TSP, decision variable x_{ij} denotes location i is visited at step j, with an additional index denoted by the vehicle number for VRP. This is the convention we adopted in the previous sections of this paper. For node-based QUBOs, the number of variables is $O(kN^2)$ for k vehicles and n locations. In the edge-based formulations, the same notation is used to describe a related but fundamentally different choice: x_{ij} denotes that the arc between location i and j is used in a tour. Then, constraints are formulated using these variables to ensure each arc is used exactly once. Thus, the number of variables required for the edge-based QUBO is $O(\mathrm{card(E)}) = O(N^2)$. Note that, in principle, upon using the edge-based formulation, it is necessary to encode constraints such that independent subtours (closed loops) cannot be present in the optimum of the problem. It is important to note that the number of subtours contained in a graph scales as $O(n!)$ and therefore it is important to deal with these constraints properly otherwise the model cannot be practically implemented. For TSP/VRP heuristics which use tour augmentation, arc insertion, or similar steps in generating solutions, this is not a bottleneck[23]; any additional arc which creates a closed loop and violates the constraints is rejected, stored, and eliminated from the candidate list of possible solutions. However, metaheuristics have no such knowledge of the problem constraints, and so all constraints need to be modelled mathematically and incorporated into the objective function directly for the QUBO. Therefore, each possible subtour must be explicitly excluded from the minimum of the objective function, otherwise the optimum becomes dominated by infeasible solutions. Thus, since subtour elimination is $O(n!)$ implementing an edge-based TSP/VRP QUBO is at least as hard as solving TSP/VRP to begin with. Therefore, to avoid subtour elimination entirely, we have to introduce an ordering of all the edges traversed by each vehicle and formulate constraints such that only valid tours are admissible. This can be done as for the node-based formulation, where each vehicle has a separate set of edge-based decision variables, and each is constrained globally (k arcs are used to leave the depot, k arcs to return, each location is connected to exactly two arcs, etc.). This results in a well-defined QUBO with at most $O(kN^3)$ variables. Obviously, this is asymptotically more variables than the node-based representation of the same problem, and so we conclude that edge-based formulations of routing problems are inappropriate for QUBO solving with metaheuristics.

The ride-pooling problem QUBO

The ride-pooling problem is another variant of routing problems that, although less studied than TSP/VRP, is extremely relevant for real-world problems[17,24,25]. We restate the definition of the RPP for clarity: given a set of ride requests and a fleet of vehicles, pick up all customers and deliver them to their respective drop-off locations while minimizing the distance traveled by the fleet. While qualitatively similar to TSP/VRP, the RPP is unique in two ways: the vehicles do not start at a depot (and are not required to return to a depot), and multiple customers can be picked up consecutively. In practice, additional constraints such as time windows, minimizing customer waiting time, minimizing vehicle deviations, and other such considerations are all included when solving RPPs.

To construct the QUBO for the RPP, we follow the methods as for VRP and TSP and start by defining the binary decision variables. As before, the variables represent the possible locations at each step of the path for each vehicle in the fleet:

$$x_{a,l,\beta} = \begin{cases} 1 & \text{vehicle} a \text{is in location} l \text{at step} \beta, \\ 0 & \text{otherwise.} \end{cases}$$

Index a enumerates the vehicles (up to A), l is the index of possible locations, and β is the total number of steps in the path for each vehicle. Since every customer has a pickup and drop-off point, then we have a maximum of $2C + 1$ steps in each vehicle's path (including the starting point), where C is the number of ride requests.

To describe the constraints of the RPP, we start with reusing as much of the formulation from VRP/TSP as possible. Let each ride request be denoted by the tuple (s_i, f_i), where s_i is the ith customer's start location (pickup) and f_i is the corresponding final location (drop-off). Therefore, each (s_i, f_i) pair corresponds to two possible locations for each vehicle in the fleet, each of which must be visited by only one vehicle. We denote this set of shared locations for vehicles by $\mathscr{L}$ (note that this does not include the starting point for any of the vehicles). We express the one-hot constraint for each request over all vehicles as follows:

$$\sum_{l\in\mathscr{L}} \left(1 - \sum_{a=1}^{A}\sum_{\beta=1}^{S} x_{a,l,\beta}\right)^2 = 0. \tag{11}$$

Note that, since we did not include the vehicles' starting points (which we will refer to as d_a) in this one-hot constraint, it is of no cost for a vehicle to "remain" in that location from one step of its path to the next. This is the same as in the VRP QUBO where vehicles can remain in their depots for as long as necessary, and for the same reason– we can't know *a priori* how many steps the optimal tour of each vehicle will be in the RPP. So, given that A is the number of vehicles and $S = 2C + 1$ is the maximum path length, we have that every location can only be visited once in the minimum of this function, excluding the vehicles' starting point.

Similarly, we constrain all steps in the path such that only one location can be visited at a time, also as in TSP/VRP. However, note that, in the case where we have more than one vehicle, the fact that we are guaranteed that at least one location is visited by any one of the vehicles means that at least one location does not need to be visited by any other vehicle. This potentially reduces the lengths of the tours of all vehicles by one step, except in the event that one vehicle must respond to all requests. Thus, we can convert the one-hot constraint corresponding to the last step in the path to a half-hot constraint, where the minimum is when exactly one or no location is visited. This has the additional consequence of making the starting location d_a of each vehicle redundant in the last step, and so we are able to reduce the number of variables in the QUBO without sacrificing feasibility, i.e. we can set $x_{a,d_a,S} = 0$ for all a. The resulting constraints are therefore a split of the summations in Eq. (9), a combination of one-hot and half-hot constraints:

$$\sum_{a=1}^{A}\sum_{\beta=1}^{S-1} \left(1 - \sum_{l\in V} x_{a,l,\beta}\right)^2 + \sum_{a=1}^{A} \left(1 - 2\sum_{l\in\mathscr{L}} x_{a,l,S}\right)^2 = 0. \tag{12}$$

Now, recall that in order for a request to be satisfied, the same vehicle visiting a pickup must also visit the corresponding drop-off. Formally, we say that a request $(s_i, f_i) \in \mathscr{C}$, where $\mathscr{C}$ is set of customer requests, is satisfied if and only if there exist a solution with vehicle $\bar{a} \in \{1, \ldots, A\}$ and steps $0 < \beta_1 < \beta_2 < S$ such that $x_{\bar{a},s_i,\beta_1} = 1$ and

$x_{\bar{a},f_i,\beta_2} = 1$. This means that the locations *s* and *f* must appear in the same vehicle's tour, and that location s_i must appear before location f_i in that tour, which we call the *causality condition*. We therefore introduce the notion of *incentive terms*: these are penalty terms in the QUBO with negative coefficients for the purpose of incentivizing combinations of binary variables in optimal solutions. To implement the causality condition in the RPP QUBO, we must incentivize all solutions where variable s_i appears before f_i for each vehicle *a*:

$$\sum_{a=1}^{A} \sum_{(s,f)\in\mathscr{C}} \sum_{\substack{\beta_1,\beta_2=1,\\ \beta_1<\beta_2}}^{S} -x_{a,s,\beta_1}\, x_{a,f,\beta_2}. \tag{13}$$

Notice that, instead of using such incentive terms, the same result could have been achieved by implementing penalty terms on all possible infeasible configurations of s_i and f_i that violate our causality condition instead. We observe that, for each pair $(s,f) \in \mathscr{C}$, incentivizing feasible or penalizing infeasible solutions yield the same desired result. This is due to the fact that for each feasible solution we know that the value of Eq. (13) is $-C$ by definition. Similarly, the set of penalty terms required to penalize all infeasible solutions is the complement to eq. (13):

$$\sum_{a=1}^{A} \sum_{(s,f)\in\mathscr{C}} \sum_{\substack{\beta_1,\beta_2=1,\\ \beta_1\geq\beta_2}}^{S} x_{a,s,\beta_1}\, x_{a,f,\beta_2} + \sum_{a_1\neq a_2} \sum_{(s,f)\in\mathscr{C}} \sum_{\beta_1,\beta_2=1}^{S} x_{a_1,s,\beta_1}\, x_{a_2,f,\beta_2},$$

which by definition evaluates to 0 for feasible solutions. Since the sets containing the terms of the two functions are complementary, and for each $(s,f) \in \mathscr{C}$ the union of the two is all combinations of binary variables in which *s* and *f* appear (together with the one-hot constraints), we therefore only require the smaller set to include in the RPP QUBO. We formalize this observation with the following proposition.

Proposition *The causality condition can be fulfilled by either incentivizing all feasible solutions or penalizing all infeasible solutions, and whichever uses fewer terms in the QUBO is sufficient.*

We note that the RPP shares many characteristics with other well-known NP-hard optimization problems formulated as QUBOs. This suggests that our proposition can be applied in general in constrained combinatorial optimization problems modeled as QUBOs and is not unique to RPP. Specifically, our causality condition exploits the property of distinguishing between feasible and infeasible solutions to a constrained optimization problem by penalization, which is more general than the routing application we apply it to. However, generalizing to a theorem requires more general definitions and assumptions over the model and therefore is out of scope for this paper.

For the objective function of the RPP, similar to TSP/VRP, we minimize the weight of the arcs between locations. In order to accommodate the modifications in Eq. (12) where d_a is removed from the last step, we modify the objective function to match:

$$\sum_{a=1}^{A} \sum_{l_1\in V} \sum_{l_2\in\mathscr{L}} \sum_{\beta=1}^{S-1} \frac{w_{l_1,l_2}}{W}\, x_{a,l_1,\beta}\, x_{a,l_2,\beta+1} + \sum_{a=1}^{A} \sum_{l\in\mathscr{L}} \sum_{\beta=1}^{S-2} \frac{w_{d,l}}{W}\, x_{a,l,\beta}\, x_{a,d_a,\beta+1}. \tag{14}$$

Here, the first addend sums the distances between every location (*including* d_a) to all other locations (*excluding* d_a) in subsequent steps in the each vehicle's path. The second addend adds the distance terms between all locations to d_a in subsequent steps, except for the last step of the tour. Note that the cost of a vehicle "staying" in d_a from one step to the other is 0. Again, by our definition of normalization factor *W*, it is never favorable to either violate constraints nor to return to d_a after leaving it. Thus, by summing Eqs. (11), (12), (13) and (14), we obtain the basic RPP QUBO formulation.

Adding real-world constraints to the RPP QUBO

To make our QUBO more representative of real-world problems, we can add additional complexity in the form of constraints or requirements which must be fulfilled in order to consider solutions feasible (or even optimal, in the case of multi-objective optimization). For example, consider that some real-world ride-hailing companies have a fleet with vehicles of different passenger capacities, or that requests can be made by groups with different numbers of people to be picked up. To solve this problem we must include the concept of capacity for vehicles in the RPP QUBO. Let each vehicle have a fixed capacity Ω_a, and each request have a number of passengers p_l that need to be collected. Then, at every step of the path, for every vehicle, the number of passengers in a vehicle cannot exceed the vehicle's capacity. This can be implemented via the inequality:

$$\sum_{l\in\mathscr{L}} \sum_{\beta=1}^{S} p_l x_{a,l,\beta} \leq \Omega_a \quad \text{for } a = 1,\ldots,A.$$

Note that $p_{s_i} = -p_{f_i}$. Meaning, the number of passengers entering the vehicle at every pickup s_i is the same as the number of passengers exiting the vehicle at f_i. To implement these inequalities in QUBO, we require a set of slack variables (denoted by $y_{a,c}$) for each vehicle. This in effect keeps a running sum at each step of the path for every vehicles ensuring that the vehicle capacity is never violated. We express the constraint as the following equality:

$$\sum_{a=1}^{A}\left(\sum_{\beta=1}^{S}\sum_{j=0}^{\beta}\sum_{l\in\mathscr{L}} p_l x_{a,l,j} - \sum_{c=1}^{\Omega_a} y_{a,c}\right)^2 = 0. \tag{15}$$

Thus, we have a complete description of the RPP including vehicle capacities. Note that many similar such real-world constraints can be implemented in similar ways, as has been done in previous literature for TSP/VRP QUBOs. We briefly present a complexity analysis of the RPP QUBO model by enumerating the number of variables needed in order to implement it. Since each vehicle is treated the same, we count the variables on each "slice" independently and then multiply by the number of vehicles A. The maximum number of steps in a vehicle's path is $2C + 1$. Therefore, the number of variables for each vehicle is $(2 \cdot C + 1)^2 - 1$, since at each step the vehicle can visit any location except d_a in the last step. Thus, the total number of variables scales as $O(A \cdot C^2)$ for the basic RPP.

Although the number of variables required for the basic RPP QUBO scales quadratically, and therefore polynomially, for quantum computing this still represents a significant overhead due to the limited number of qubits that are available in state-of-the-art processors at the time of writing. Indeed, for each variable in the QUBO model we need at least one qubit in the quantum computer. Hence, finding ways to lower the number of variables in the QUBO can have an impact on quantum algorithm performance. By relying on some intuition from the description of the RPP, we can remove some redundant variables by fixing their values *a priori*. For example, each vehicle has a unique starting point, which implies that no other location can be the first step in the path. This means that, for all locations $\mathscr{L}$ we can fix the variables with $\beta = 1$ as follows:

$$x_{a,d_a,1} = 1 \text{ and } x_{a,l,1} = 0 \quad \text{for all } a = 1, \ldots, A \text{ and } l \in \mathscr{L}.$$

Furthermore, by analyzing constraints in Eq. (13), we see that it is impossible to go from d_a directly to a drop-off. Furthermore, it is impossible to end a path at any pickup. We can fix these variables as well:

$$x_{a,s,S} = 0 \text{ and } x_{a,f,2} = 0 \quad \text{for all } a = 1, \ldots, A \text{ and } (s,f) \in \mathscr{C}.$$

Although this is a modest (linear) improvement in the number of variables used, adopting this in practice would allow practitioners to solve larger sets of RPP QUBOs than without this method. However, to make these QUBOs more representative of real-world problems, we must also include additional constraints in the problem, and account for them in the number of variables in the QUBO. Here we use the capacities of the vehicles as an example. The number of QUBO variables increases by the number of slack variables needed to represent the capacity constraints with equalities. By definition of slack variables, they must sum to the capacity of the vehicle, and hence we require Ω_a variables for each step in the path. Thus, we need at most $O(A\mathscr{P}\ C)$ additional QUBO variables for all capacity constraints, where $\mathscr{P} := \max_{a=1,\ldots,A} \Omega_a$. In total, the number of variables for the RPP QUBO with capacity constraints scales as $O(A[C^2 + \mathscr{P}\ ])$.

Conclusions

In this work, we presented several implementations of different routing problems considering and explaining for each of these its QUBO formulation. We analyze the different formulations discussing their implementation and complexity cost. Furthermore, we stressed how proposed edge-based formulations of routing problems were incorrectly implemented, yielding an exponential number of terms in the QUBO model. We also proposed a technique that leads to a model that can be implemented in polynomial time. In addition, we noted that the QUBO model of the edge-based formulation always requires more variables and terms than the node-based formulations.

The main contribution of this work is a new routing problem implementation called the Ride Pooling Problem (RPP), which can be used to model several dial-a-ride problems of goods, people or services. This problem presents new challenges compared to other routing problems since it requires additional constraints for visiting the locations. This was done using the novel formalism of incentive terms that, combined with the classical penalty terms used in QUBO models, gives an efficient implementation that reduces the number of terms in the model. With this, we could identify redundant terms that do not add features to the model and can be set to zero, reducing the number of terms. These were found by analyzing the constraints of the original problem and defining the relationship between the constraints properties and the role of the variables in the QUBO model. This has strong implications in quantum optimization in particular, since most quantum optimization algorithms scale with the number of terms of the QUBO. Furthermore, we show how to implement a complicated form of capacity constraint, that describes the property of the vehicles in this model to be loaded or unloaded during the route.

Despite the implementation of the proposed model is efficient, we can note that the quadratic scaling of the number of variables in the model is not favourable, in particular in the field of quantum optimization, where the specific devices devoted to solving optimization problems can be considered small and no large-size instance can be solved on them. Moreover, when the capacity constraints are considered, the number of variables and terms makes this QUBO model not tractable with quantum devices even for small sizes. This is due to the use of slack variables to implement the inequalities, which significantly increase the number of variables that must be used to implement this model. Therefore, further developments in formulating combinatorial optimization problems as QUBO models are required to solve these problems with quantum computers. In particular, the formulation of constraints is a long-lasting problem that prevents the implementation of arbitrary optimization problems.

Data availability

All data generated or analysed during this study are included in this published article.

Received: 19 February 2024; Accepted: 20 August 2024
Published online: 26 August 2024

References

1. Kochenberger, G. *et al.* The unconstrained binary quadratic programming problem: A survey. *J. Comb. Optim.* **28**, 58–81. https://doi.org/10.1007/s10878-014-9734-0 (2014).
2. Lucas, A. Ising formulations of many np problems. *Front. Phys.* **2**, 5 (2014).
3. Farhi, E., Goldstone, J., Gutmann, S. & Sipser, M. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106* (2000).
4. Farhi, E., Goldstone, J. & Gutmann, S. A quantum approximate optimization algorithm. *arXiv preprint*[SPACE]arXiv:1411.4028 (2014).
5. Kadowaki, T. & Nishimori, H. Quantum annealing in the transverse ising model. *Phys. Rev. E* **58**, 5355 (1998).
6. Johnson, M. W. *et al.* Quantum annealing with manufactured spins. *Nature* **473**, 194–198 (2011).
7. Arute, F. *et al.* Quantum supremacy using a programmable superconducting processor. *Nature* **574**, 505–510 (2019).
8. Li, G. *et al.* On the co-design of quantum software and hardware. In: *Proc. Eight Annual ACM International Conference on Nanoscale Computing and Communication*, 1–7 (2021).
9. Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **41**, 303–332 (1999).
10. Feynman, R. P. Simulating physics with computers. *Int. J. Theor. Phys.***21** (1982).
11. Yarkoni, S., Raponi, E. & Bäck, T. & Schmitt, S. Introduction and review. Reports on Progress in Physics, Quantum annealing for industry applications. (2022).
12. Streif, M., Yarkoni, S., Skolik, A., Neukart, F. & Leib, M. Beating classical heuristics for the binary paint shop problem with the quantum approximate optimization algorithm. *Phys. Rev. A* **104**, 012403. https://doi.org/10.1103/PhysRevA.104.012403 (2021).
13. Yarkoni, S. *et al.* Multi-car paint shop optimization with quantum annealing. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 35–41, https://doi.org/10.1109/QCE52317.2021.00019 (2021).
14. Harrigan, M. P. *et al.* Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. *Nat. Phys.* **17**, 332–336 (2021).
15. Barahona, F. On the computational complexity of ising spin glass models. *J. Phys. A Math. Gen.* **15**, 3241 (1982).
16. Karp, R. M. *Reducibility among Combinatorial Problems* 85–103 (Springer, 1972).
17. Ke, J., Yang, H. & Zheng, Z. On ride-pooling and traffic congestion. *Transp. Res. B Methodol.* **142**, 213–231 (2020).
18. Feld, S. *et al.* A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer. *Front. ICT* **6**, 13. https://doi.org/10.3389/fict.2019.00013 (2019).
19. Borowski, M. *et al.* New hybrid quantum annealing algorithms for solving vehicle routing problem. In *Computational Science - ICCS 2020* (eds Krzhizhanovskaya, V. V. *et al.*) 546–561 (Springer International Publishing, 2020).
20. Papalitsas, C., Andronikos, T., Giannakis, K., Theocharopoulou, G. & Fanarioti, S. A qubo model for the traveling salesman problem with time windows. *Algorithms* **12**, 224. https://doi.org/10.3390/a12110224 (2019).
21. Irie, H., Wongpaisarnsin, G., Terabe, M., Miki, A. & Taguchi, S. Quantum annealing of vehicle routing problem with time, state and capacity. In *Quantum Technology and Optimization Problems: First International Workshop, QTOP 2019, Munich, Germany, March 18, 2019, Proceedings 1* (ed. Irie, H.) 145–156 (Springer, 2019).
22. Christofides, N. *Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem* (Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976).
23. Dantzig, G., Fulkerson, R. & Johnson, S. Solution of a large-scale traveling-salesman problem. *J. Oper. Res. Soc. Am.* **2**, 393–410 (1954).
24. Molenbruch, Y., Braekers, K. & Caris, A. Typology and literature review for dial-a-ride problems. *Ann. Oper. Res.* **259**, 295–325 (2017).
25. Ho, S. C. *et al.* A survey of dial-a-ride problems: Literature review and recent developments. *Transp. Res. B Methodol.* **111**, 395–421 (2018).

Acknowledgements

MC and SY are funded by the German Ministry for Education and Research (BMB+F) in the project QAI2-Q-KIS under grant 13N15587. Furthermore, the authors would like to thank Gabriele Compostella, Anestis Papanikolaou, Andrea Skolik and Anton Suchaneck for the valuable discussions and suggestions given.

Author contributions

The model development for the ride pooling problem is shared equally between MC and SY. MC made a major contribution to the theoretical results. The statements and the final derivations of the theoretical results are equally distributed between the authors. MC and SY equally contributed to the writing of the work.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to M.C.