IEEE Transactions on Quantum Engineering

Received October 29, 2020; revised December 16, 2020; accepted December 19, 2020; date of publication January 6, 2021; date of current version February 4, 2021.

Digital Object Identifier 10.1109/TQE.2021.3049230

Formulating and Solving Routing Problems on Quantum Computers

STUART HARWOOD[1], CLAUDIO GAMBELLA[2], DIMITAR TRENEV[1], ANDREA SIMONETTO[2] (Member, IEEE), DAVID BERNAL[3], AND DONNY GREENBERG[4]

[1] Corporate Strategic Research, ExxonMobil Research and Engineering, Annandale, NJ 08801 USA
[2] IBM Quantum, IBM Research Ireland, Dublin 15, Ireland
[3] Quantum Computing Group, Carnegie Mellon University, Pittsburgh, PA 15213 USA
[4] IBM Quantum, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA

(Stuart Harwood, Claudio Gambella, and Dimitar Trenev contributed equally to this work.) Corresponding author: Claudio Gambella (claudio.gambella1@ie.ibm.com).

ABSTRACT The determination of vehicle routes fulfilling connectivity, time, and operational constraints is a well-studied combinatorial optimization problem. The NP-hard complexity of vehicle routing problems has fostered the adoption of tailored exact approaches, matheuristics, and metaheuristics on classical computing devices. The ongoing evolution of quantum computing hardware and the recent advances of quantum algorithms (i.e., VQE, QAOA, and ADMM) for mathematical programming make decision-making for routing problems an avenue of research worthwhile to be explored on quantum devices. In this article, we propose several mathematical formulations for inventory routing cast as vehicle routing with time windows and comment on their strengths and weaknesses. The optimization models are compared from a quantum computing perspective, specifically with metrics to evaluate the difficulty in solving the underlying quadratic unconstrained binary optimization problems. Finally, the solutions obtained on simulated quantum devices demonstrate the relative benefits of different algorithms and their robustness when put into practice.

INDEX TERMS Optimization, quantum computing, routing, variational algorithms.

I. INTRODUCTION

Routing problems encompass a wide range of problems in logistics and operations research. These problems are generally concerned with the optimal management of a fleet of vehicles, e.g., how should each vehicle be dispatched in order to satisfy some goal, while minimizing time or maximizing profit. There are many variants and specifications of the problem to certain settings [1], and this work focuses on the vehicle routing problem with time windows (VRPTW). Typical solution approaches to VRPTW on classical computing devices include mathematical formulations involving discrete variables [2]. Consequently, classical methods tend to have worst-case solution times that scale exponentially with the number of decision variables (the 0–1 integer programming feasibility problem is NP-complete [3, Sec. I.5, Prop. 6.6]).

For specific applications, tailored exact approaches, matheuristics, and metaheuristics need to be devised with the aim of obtaining solutions with good quality at a reasonable computational effort. For example, we are motivated by the maritime inventory routing problem (MIRP) [4], in which inventory levels of a product must be tracked. This is a characteristic of commodity and bulk (e.g., oil, gas, iron ore, and grain) shipping, which in 2017 accounted for over 75% of world seaborne trade measured in ton-miles [5, Fig. 1.4]. Recent work in [6] indicates the limits of some of these classical methods for the MIRP.

The ongoing evolution of quantum computing hardware and the recent advances of quantum algorithms for mathematical programming make decision-making for routing problems an avenue of research worthwhile to be explored on quantum devices. So far, quantum algorithms for mathematical optimization have often focused on quadratic unconstrained binary optimization (QUBO) problems [7], [8], expressed in the form

$$\min_{\mathbf{x}} \ \mathbf{x}^\top \mathbf{M}\mathbf{x} + c \qquad \text{s.t. } \mathbf{x} \in \{0,1\}^n \tag{1}$$

where $\mathbf{M}$ is an $n \times n$ real matrix and c is a scalar offset. To achieve a representation on quantum devices, the QUBO can be transformed into an Ising model with Hamiltonian constituted as a summation of weighted tensor products of

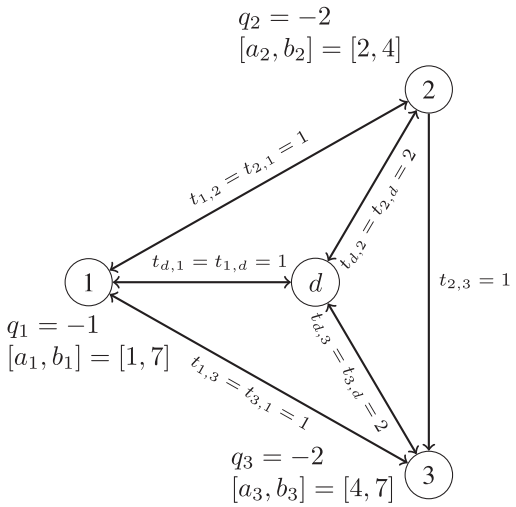


FIGURE 1. Small example VRPTW. Arc costs $c_{i,j}$ equal the travel time $t_{i,j}$. This is a nearly fully connected graph; however, travel from node 3 to node 2 is not allowed because the time window of node 2 ends before the time window of node 3 begins. The vehicles leave depot d fully loaded at their capacity $Q = 6$. Recall the sign convention for q_i; q_i is negative for demand that must be delivered and depletes the product on a vehicle.

Z Pauli operators. Equality constraints can also be added to the QUBO formulation by casting them as a quadratic penalization of the objective function [9], [10]. One of the goals of our work is then to introduce formulations of the routing problem that are amenable to emerging quantum hardware and quantum algorithms, which can provide fast heuristics for QUBO problems.

Gate-based quantum computers provide a particularly alluring setting to develop and propose new algorithms. Variational algorithms such as the quantum approximate optimization algorithm (QAOA) [11] and variational quantum eigensolver (VQE) [12] can be implemented on current gate-based devices. The idea behind quantum variational algorithms is to use quantum devices to efficiently sample from large parameterized distributions by executing shallow parameterized quantum circuits. In an optimization setting, when solving QUBO problems, the quantum device is used to efficiently calculate a utility function (e.g., the expectation value or conditional value at risk [7]) of the cost Hamiltonian for a particular parameterized quantum state. A classical optimization algorithm is used to optimize the given utility function over the rotation parameters. Provided that the parameterized circuit, also called an *ansatz*, represents states close to the Hamiltonian ground state, variational algorithms can obtain heuristic solutions of reasonable quality on current quantum devices. There are several reasons to motivate the adoption of quantum algorithms and solve QUBO problems using quantum computers. First, variational algorithms such as VQE and QAOA are known to not be efficiently simulatable classically [13], [14]. Second, some encouraging results in terms of performance advantage have been documented for QAOA with respect to the classical Goemans–Williamson limit [15], for Grover Adaptive Search with respect to a classical unstructured search [16], and for quantum semidefinite programming relaxations for QUBO problems [17].

Decision-making in practical optimization problems often involves modeling continuous variables and inequality constraints with mixed binary optimization (MBO) [18]. A recent contribution to solving MBO on current quantum devices is given by nonconvex variants of the alternating direction method of multipliers (ADMM). Specifically, the ADMM works by alternatively optimizing an augmented Lagrangian function over QUBO and continuous subproblems. The QUBO problem can be solved on quantum devices via quantum algorithms such as QAOA and VQE. Conditions of convergence to stationary points have been investigated, and the ADMM has been shown to achieve satisfactory levels of solution quality [19].

This article aims to bring together different formulations for solving routing problems on quantum devices, with specific focus on timing constraints, expressed either in a discrete or continuous form. These formulations are largely inspired by the classical operations research literature, but we introduce them here in order to obtain QUBO representations suitable for quantum algorithms. Previous work on quantum or quantum-inspired classical algorithms applied to logistics problems has focused on the traveling salesman problem [8], [20], [21] or the related capacitated vehicle routing problem [22]. While Irie *et al.* [23] have introduced a formulation that handles timing constraints, we are not aware of any work that collects VRPTW approaches together and compares them in the context of quantum algorithms (see [24] for a comparison of different formulations in a classical setting). Overall, the contributions of this article are the following.

1) We introduce mathematical optimization models for VRPTW suitable for state-of-the-art quantum algorithms.
2) We investigate the modeling capabilities of the VRPTW formulations with respect to constraints and objectives arising in practical applications.
3) We compare the VRPTW formulations from a quantum computing perspective, specifically with metrics to evaluate the difficulty in solving the underlying QUBO problems.
4) We test the formulations on simulated gate-based quantum devices and draw insights on the current capabilities of quantum computing approaches for routing problems.

Given that the quantum algorithms proposed here for the VRPTW are not bound to a specific quantum QUBO solver, our framework is flexible for future improvements in both gate-based quantum hardware and algorithms, as well as other types of emerging specialized devices. This is especially appealing, given the wealth of studies and results in this direction (see, e.g., [25]–[29]).

IEEE Transactions on Quantum Engineering

The rest of this article is organized as follows.

1) Section II describes the VRPTW problem that we consider, along with the key features of the MIRP (see Section II-A) that can be cast in the VRPTW formalism. The various mathematical programming formulations of the VRPTW are introduced in Sections II-B–II-E.
2) In Section III, we discuss how to cast and solve the VRPTW formulations on quantum devices, via reformulations to QUBO problems (see Sections III-A and III-B) or an MBO problem (Section III-C).
3) Section IV compares the formulations along different dimensions on the examples presented in Section IV-A. In particular, the strengths and weaknesses of the formulations in modeling VRPTW are discussed in Section IV-B, while the difficulty for solving the formulations is evaluated in Sections IV-C and IV-D via the number of variables, sparsity of the QUBO matrix, and the number of solutions.
4) Section IV-E includes numerical results obtained from solving a VRPTW instance with the outlined quantum algorithms. Sequence-based formulation and sequence-based formulation with continuous time are solved in the section.
5) Finally, conclusions and remarks are drawn in Section V.

Notation: Throughout this article, we use the following notation. Sets are denoted with uppercase calligraphic letters (e.g., $\mathcal{A}$). Matrices are denoted with uppercase bold letters (e.g., $\mathbf{M}$), while vectors are lowercase bold letters (e.g., $\mathbf{v}$). For a vector $\mathbf{v}$, $\mathbf{Diag}(\mathbf{v})$ denotes a square matrix with $\mathbf{v}$ on its diagonal and zeros elsewhere.

II. MATHEMATICAL FORMULATIONS FOR VRPTW

In this section, we describe the VRPTW at hand and specify the different mathematical formulations. An instance of VRPTW is formulated on a graph with nodes $\mathcal{N} \cup \{d\}$ and directed edges/arcs $\mathcal{A}$. Each node $i \in \mathcal{N}$ represents a customer, associated with a demand level q_i and time window $[a_i, b_i]$: q_i represents the amount of product that must be delivered ($q_i < 0$) or picked up ($q_i > 0$) after time a_i and before time b_i. The "depot" node d serves as departure and destination node for all vehicles $v \in \mathcal{V}$. In some situations, the depot node may be considered a physical node corresponding to, for instance, a warehouse, from which all vehicles start their routes fully loaded, and at which all vehicles must finish their routes. We note that the formulations discussed in this work could be seamlessly extended to the case in which starting and ending nodes for the vehicle routes are not coincident with the depot. We allow a vehicle to arrive early (i.e., before a_i) and wait at a customer location, but not to arrive late (i.e., after b_i). Each customer is serviced exactly once (i.e., the demand cannot be split among different vehicles.). Each arc $(i, j) \in \mathcal{A}$ has an associated cost $c_{i,j}$ and travel time $t_{i,j}$. Typically, the cost for traveling from node i to j is the distance between the nodes. The vehicles are homogeneous (i.e., they have the same capacity, travel speed, cost to operate, etc.). Each vehicle has capacity (maximum load size) Q and leaves the depot with an initial loading Q_0. For instance, when the depot corresponds to a warehouse, we might assume $Q_0 = Q$. If it is ever needed, it is assumed that the depot has an infinite time window $[0, +\infty]$. The objective of the VRPTW is to minimize the total cost of transportation while servicing each customer in the given time windows.

In Section II-A, we describe some key features of the MIRP and how these manifest in the data of the VRPTW described above. A comprehensive overview of the MIRP and a review of the literature can be found in [4]. In the following subsections, we describe four different mathematical formulations of the VRPTW: route-based, arc-based, sequence-based, and sequence-based with continuous time. Since our goal is to target QUBO subproblems, these formulations have primarily or exclusively binary variables and linear or quadratic equality constraints. The exception is the sequence-based formulation with continuous time in Section II-E, which involves inequality constraints and continuous variables. These formulations have different levels of faithfulness in modeling VRPTW, and we discuss extensions that could make formulations more accurately capture the VRPTW setting.

A. KEY FEATURES OF THE MIRP

The MIRP is often characterized by travel times that are relatively long compared to the time windows of the nodes. These long travel times mean that fairly long time horizons must be considered in order to get the benefit of optimizing logistics. Furthermore, there are often multiple supply points in addition to multiple demand points/customers. These supply points are often terminals producing a commodity such as natural gas, and since they often have limited storage capacity, they must also be serviced, and inventory picked up, in certain time intervals. This is why demand levels q_i are signed. Another characteristic of maritime shipping, in particular in liquefied natural gas shipping, is that vessels/vehicles typically fully load at supply points and fully unload at demand points. Therefore, a typical route of a vessel alternates between supply points and demand points. Combined with the assumption of a homogeneous fleet of vessels, this means that the capacity Q and the demand levels q_i are all equal in magnitude. Consequently, the constraints on vehicle capacity are less important. It is instead more important to enforce the alternating sequence of supply and demand points. This is easily achieved by restricting the arcs in $\mathcal{A}$ so that there are only arcs between a supply and demand node or *vice versa*. Combined with the long travel times, narrow time windows, and long time horizons, the graph can be quite sparse, as we can *a priori* remove arcs that would have the vessel arriving at the node after the time window ends.

B. ROUTE-BASED FORMULATION

In the route-based VRPTW formulation, the decisions to be made are whether routes are traveled or not. A route is a sequence of nodes $(i_1, i_2, \ldots, i_P)$ satisfying the following constraints (for some route-specific positive integer P). A route begins and ends at the depot: $i_1 = i_P = d$. Each segment is a valid arc: $(i_p, i_{p+1}) \in \mathcal{A}$, for all $1 \leq p \leq P-1$. For each feasible route, the running sum of the product delivered/picked up must be physically possible: $0 \leq Q_0 + \sum_{j=2}^{p} q_{i_j} \leq Q$, for all $p \leq P-1$ (i.e., at each stop, the amount loaded on the vehicle must be nonnegative and less than the vehicle's capacity). The arrival time at node i_p must be before the time window ends. If we let $T_{i_1} = 0$, then the effective arrival time at node i_{p+1} is given by $T_{i_{p+1}} = \max\{a_{i_{p+1}}, T_{i_p} + t_{i_p, i_{p+1}}\}$ for all $1 \leq p \leq P-1$. Then, we require $T_{i_p} \leq b_{i_p}$ for all p.

We index the set of routes by the set $\mathcal{R}$. If route $r \in \mathcal{R}$ has node sequence $(i_1, i_2, \ldots, i_P)$, then it has cost $c_r = \sum_{p=1}^{P-1} c_{i_p, i_{p+1}}$. Finally, we define $\delta_{i,r}$ to be a constant with value 1 if route r visits customer $i \in \mathcal{N}$ (i.e., one of the nodes in the route is i), and zero otherwise.

We introduce variable x_r, which has value 1 if a vehicle travels route r; otherwise, it has value 0. Here, and in the rest of this section, we denote the collection of binary variables in a specific formulation by $\mathbf{x}$. In the case of the route-based formulation, $\mathbf{x} = (x_r)_{r \in \mathcal{R}}$. As a mathematical program, the VRPTW becomes

$$\min_{\mathbf{x}} \quad \sum_{r \in \mathcal{R}} c_r x_r \tag{2a}$$

$$\text{s.t.} \quad \sum_{r \in \mathcal{R}} \delta_{i,r} x_r = 1 \quad \forall i \in \mathcal{N} \tag{2b}$$

$$x_r \in \{0, 1\} \quad \forall r \in \mathcal{R}. \tag{2c}$$

The equality constraint enforces the requirement that all customer nodes are visited by exactly one vehicle. A constraint on the number of vehicles available can be enforced by making sure that the number of outgoing arcs from the depot d equals the number of available vehicles $|\mathcal{V}|$. If necessary, the only nodes directly connected to the depot can be thought of as "dummy" nodes and essentially keep track of whether a vehicle is used or not. Problem (2) may also be recognized as a set partitioning problem/exact covering problem, a classic problem in discrete optimization; see, for instance, [21, Sec. 4.1] and [30].

The number of routes $|\mathcal{R}|$ can, in general, be extremely large. If travel from any node to any other node is possible, then the number of routes is $|\mathcal{N}|!$, although the allowed arcs and constraints on a route described above will limit this. In classical computing approaches, this formulation is best handled by a column generation method (e.g., [30]). Here, we will consider $\mathcal{R}$ to be given. The instances that we will consider are either small enough that the routes can be exhaustively enumerated, or else we will use a heuristic to generate a set of routes (see Algorithm 1 in Appendix V). While this heuristic is not computationally expensive, it is effectively a preprocessing step that influences the characteristics of the formulation. However, this enables us to test the formulation on current quantum devices with quantum optimization algorithms.

Meanwhile, this motivates the rest of the formulations in this section, which can be fully specified without having to perform this potentially expensive route generation step.

C. ARC-BASED FORMULATION

We now consider a discrete-time arc-based formulation of the VRPTW. In this case, the range of time instants, in which the vehicles can perform their routes, is represented by the discrete set $\mathcal{T}$. Each node's time window encloses at least one time point in $\mathcal{T}$: $\mathcal{T} \cap [a_i, b_i] \neq \emptyset$.

We introduce a variable $x_{i,s,j,t}$, which has value 1 if a vehicle travels from node i at time s to node j at time t; otherwise, it has value 0. As a mathematical program, the VRPTW becomes

$$\min_{\mathbf{x}} \quad \sum_{i,s,j,t} c_{i,j} x_{i,s,j,t} \tag{3a}$$

$$\text{s.t.} \quad \sum_{i,s,t} x_{i,s,j,t} = 1 \quad \forall j \in \mathcal{N} \tag{3b}$$

$$\sum_{j,t} x_{j,t,i,s} = \sum_{j,t} x_{i,s,j,t} \quad \forall (i,s) \in \mathcal{N} \times \mathcal{T} \tag{3c}$$

$$x_{i,s,j,t} = 0 \quad \forall (i,s,j,t) : t \notin [a_j, b_j] \tag{3d}$$

$$x_{i,s,j,t} = 0 \quad \forall (i,s,j,t) : s \notin [a_i, b_i] \tag{3e}$$

$$x_{i,s,j,t} = 0 \quad \forall (i,s,j,t) : (i,j) \notin \mathcal{A} \tag{3f}$$

$$x_{i,s,j,t} = 0 \quad \forall (i,s,j,t) : s + t_{i,j} > t \tag{3g}$$

$$x_{i,s,j,t} \in \{0, 1\} \quad \forall (i,s,j,t). \tag{3h}$$

Constraints (3b) ensure that each node (besides the depot d) is visited exactly once over all vehicles. Constraints (3c) ensure continuity of routes through the graph: if a vehicle enters node i at time s, then it must leave node i at that time. Note that we do not enforce this for the depot d; otherwise, the vehicles could not get started. Constraints (3d) ensure that a vehicle arrives at a node before or during its time window. Constraints (3e) are implied by (3c) and (3b), but explicitly, a vehicle must leave a node during its time window. Constraints (3f) ensure that vehicles obey the allowed travel arcs. Constraints (3g) ensure that vehicles do not travel back in time. Note that we do not enforce the timing "exactly"; we allow $x_{i,s,j,t}$ with $s + t_{i,j}$ strictly less than t. This permits the situation that a vehicle arrives at node j at time $s + t_{i,j}$, waits, and then leaves at time t. (Thus, it is more accurate to say that variable $x_{i,s,j,t}$ takes value 1 if a vehicle leaves node i at time s, travels to node j, and then subsequently leaves node j at time t.)

Note that capacity constraints on the vehicles have not been enforced in this formulation. To do so, we could modify the formulation by adding an index $v \in \mathcal{V}$ to track the vehicles used. Constraint (3b) would be modified to

IEEE Transactions on Quantum Engineering

make sure that each node is visited exactly once over all vehicles: $\sum_{v,i,s,t} x_{v,i,s,j,t} = 1$ for each $j \in \mathcal{N}$. Meanwhile, constraint (3c) would be modified to enforce continuity of routes for each vehicle: $\sum_{j,t} x_{v,j,t,i,s} = \sum_{j,t} x_{v,i,s,j,t}$, for each $(v, i, s) \in \mathcal{V} \times \mathcal{N} \times \mathcal{T}$. The other constraints are modified similarly. Then, capacity constraints can be enforced with

$$0 \leq Q_0 + \sum_{i,s,j,u:u\leq t} q_j x_{v,i,s,j,u} \leq Q \quad \forall (v,t) \in \mathcal{V} \times \mathcal{T}$$

(the cumulative loaded product up to time t must be nonnegative and less than the vehicle's capacity). This is similar to the constraint on routes from Section II-B. This prevents an initially empty vehicle with capacity 2 from visiting three supply nodes with demand level 1 each, and then two nodes with demand level -1 each; while the sum of the loadings is 1, the capacity of the vehicle was exceeded at its third stop. However, as described in Section II, the capacity constraints are less important for practically modeling MIRP. Hence, in the numerical tests, we leave these constraints out. Furthermore, we will see that this formulation will typically be the largest in terms of number of variables, and adding an index to track specific vehicles will only exacerbate that.

D. SEQUENCE-BASED FORMULATION

We describe a discrete sequence-based formulation for the VRPTW. For this formulation, each vehicle can make a maximum number of stops P: this bound can be determined from capacity limitations, by application requirements, or simply by the number of customers. Since each vehicle starts and ends at the depot, $P - 2$ is the maximum number of nondepot nodes that each vehicle can visit. Furthermore, we assume that the depot is "absorbing": if a vehicle returns to the depot, it must remain there. Consequently, the arc (d, d) is in the arc set $\mathcal{A}$.

We introduce a variable $x_{v,p,i}$, which has value 1 if vehicle v visits node i at position p in its sequence; otherwise, it has value 0. This makes a discretization of the time horizon unnecessary to model the VRPTW routes. As a math program, the VRPTW becomes

$$\min_{\mathbf{x}} \sum_{v} \sum_{p<P} \sum_{(i,j)\in\mathcal{A}} c_{i,j} x_{v,p,i} x_{v,p+1,j} \tag{4a}$$

$$\text{s.t.} \sum_{v\in V} \sum_{p=1}^{P} x_{v,p,i} = 1 \quad \forall i \in \mathcal{N} \tag{4b}$$

$$\sum_{i\in\mathcal{N}\cup\{d\}} x_{v,p,i} = 1 \quad \forall v \in \mathcal{V}, p \in \{1,\ldots,P\} \tag{4c}$$

$$x_{v,p,i} x_{v,p+1,j} = 0 \quad \forall v \in \mathcal{V} \quad p \in \{1,\ldots,P-1\}, (i,j) \notin \mathcal{A} \tag{4d}$$

$$x_{v,p,d} x_{v,p+1,j} = 0 \quad \forall v \in \mathcal{V} \quad p \in \{2,\ldots,P-1\}, j \neq d \tag{4e}$$

$$x_{v,1,d} = 1 \quad \forall v \in \mathcal{V} \tag{4f}$$

$$x_{v,P,d} = 1 \quad \forall v \in \mathcal{V} \tag{4g}$$

$$x_{v,1,i} = 0 \quad \forall v \in \mathcal{V}, i \in \mathcal{N} \tag{4h}$$

$$x_{v,P,i} = 0 \quad \forall v \in \mathcal{V}, i \in \mathcal{N} \tag{4i}$$

$$x_{v,2,j} = 0 \quad \forall v \in \mathcal{V}, (d,j) \notin \mathcal{A} \tag{4j}$$

$$x_{v,P-1,j} = 0 \quad \forall v \in \mathcal{V}, (j,d) \notin \mathcal{A} \tag{4k}$$

$$x_{v,p,i} \in \{0,1\} \quad \forall (v,p,i).$$

Constraints (4b) ensure that each node is visited exactly once over all vehicles and sequence positions (besides the depot node d). Constraints (4c) ensure that each vehicle uses each position p in the sequence once. Constraints (4d) ensure that only allowed edges are traversed. Constraints (4e) ensure that once a vehicle returns to the depot, it remains there (i.e., it does not visit other nodes). Constraints (4f) and (4g) ensure that all vehicles start and end at the depot, respectively.

Constraints (4h)–(4k) are consequences of the others and help eliminate variables. Specifically, constraints (4h) and (4j) are a consequence of the assumption that all vehicles start at the depot: constraints (4h) are implied by constraints (4c) and (4f), while constraints (4j) are implied by constraints (4d) and (4f). Similarly, constraints (4i) and (4k) are a consequence of the assumption that all vehicles end at the depot. We may eliminate further variables by considering the nodes reachable by routes of a given length, but as discussed in Section II-B, the full set of routes can be expensive to find.

Capacity constraints on the vehicles are not directly enforced in this formulation either, in the general case. If all nodes $i \in \mathcal{N}$ have the same level of demand, capacity constraints can be enforced by choosing P appropriately. Capacity constraints can also be handled more exactly by adding the constraints

$$0 \leq Q_0 + \sum_{r=2}^{p} \sum_{i} q_i x_{v,r,i} \leq Q$$

$$\forall (v,p) \in \mathcal{V} \times \{1,\ldots,P-1\}.$$

However, as discussed before, capacity constraints are less important in the maritime setting, so this modeling feature is not included in the numerical testing.

Note that timing constraints have not been addressed explicitly in this formulation. In many problems, the time windows of the nodes are fairly narrow compared to the travel times between nodes and overall time horizon of the problem. Consequently, we can conservatively enforce the time windows by removing arcs from the graph that do not satisfy application-specific assumptions. Specifically, if the end of the time window of node i plus the travel time $t_{i,j}$ is greater than the end of the time window of node j, then that arc is removed. This corresponds to assuming that the vehicles always arrive at a node at the end of its time window. Thus, the set of arcs $\mathcal{A}$ used in this formulation may have to be different from the one used in the other formulations. For some problems, this may be too restrictive, which motivates the formulation of the following section.

E. SEQUENCE-BASED FORMULATION WITH CONTINUOUS TIME

We here consider a sequencing-based formulation of the VRPTW with continuous variables to track the arrival time at each node. This makes it possible to seamlessly enforce time windows, unlike the formulation with binary decisions only, described in Section II-D.

As in the discrete sequence-based formulation, we define a variable $x_{v,p,i}$, which has value 1 if vehicle v visits node i at position p in its sequence; otherwise, it has value 0. Furthermore, we introduce a variable $s_i \in \mathbb{R}$, which equals the time when a vehicle arrives at node $i \neq d$. The arrival time of each vehicle v to the destination node d is the duration of route v and is associated with variable s_v^d. These real variables are collected as a vector $\mathbf{s}$. As a math program, the VRPTW becomes

$$\min_{\mathbf{x},\mathbf{s}} \sum_{v} \sum_{p<P} \sum_{(i,j)\in\mathcal{A}} c_{i,j} x_{v,p,i} x_{v,p+1,j} \tag{5a}$$

$$\text{s.t.} \sum_{v\in\mathcal{V}} \sum_{p=1}^{P} x_{v,p,i} = 1 \quad \forall i \in \mathcal{N} \tag{5b}$$

$$\sum_{i\in\mathcal{N}\cup\{d\}} x_{v,p,i} = 1 \quad \forall v \in \mathcal{V}, p \in \{1,\ldots,P\} \tag{5c}$$

$$x_{v,p,i} x_{v,p+1,j} = 0 \quad \forall v \in \mathcal{V} \quad p \in \{1,\ldots,P-1\}, (i,j) \notin \mathcal{A} \tag{5d}$$

$$x_{v,p,d} x_{v,p+1,j} = 0 \quad \forall v \in \mathcal{V} \quad p \in \{2,\ldots,P-1\}, j \neq d \tag{5e}$$

$$x_{v,1,d} = 1 \quad \forall v \in \mathcal{V} \tag{5f}$$

$$x_{v,P,d} = 1 \quad \forall v \in \mathcal{V} \tag{5g}$$

$$x_{v,1,i} = 0 \quad \forall v \in \mathcal{V}, i \in \mathcal{N} \tag{5h}$$

$$x_{v,P,i} = 0 \quad \forall v \in \mathcal{V}, i \in \mathcal{N} \tag{5i}$$

$$x_{v,2,j} = 0 \quad \forall v \in \mathcal{V}, (d,j) \notin \mathcal{A} \tag{5j}$$

$$x_{v,P-1,j} = 0 \quad \forall v \in \mathcal{V}, (j,d) \notin \mathcal{A} \tag{5k}$$

$$s_i \geq \sum_{v\in\mathcal{V}} \sum_{p<P} \sum_{j:(j,i)\in\mathcal{A}} (s_j + t_{j,i}) x_{v,p,j} x_{v,p+1,i} \quad \forall i \in \mathcal{N} \tag{5l}$$

$$s_v^d \geq \sum_{p<P} \sum_{j:(j,d)\in\mathcal{A}} (s_j + t_{j,d}) x_{v,p,j} x_{v,p+1,d} \quad \forall v \in \mathcal{V} \tag{5m}$$

$$a_i \leq s_i \leq b_i \quad \forall i \in \mathcal{N} \tag{5n}$$

$$x_{v,p,i} \in \{0,1\} \quad \forall (v,p,i)$$

$$s_i \in \mathbb{R} \quad \forall i \in \mathcal{N}$$

$$s_v^d \in \mathbb{R} \quad \forall v \in \mathcal{V}.$$

All constraints pertaining only to the binary variables $x_{v,p,i}$ are shared with the sequence-based formulation discussed in Section II-D. Constraints (5n) enforce the time window constraints explicitly. Constraints (5l) define the arrival times at nodes $i \in \mathcal{N}$ and can be justified as follows. Constraints (5b) and the fact that the x variables are binary imply that for each i, there is exactly one index (v^i, p^i) such that $x_{v^i,p^i,i} = 1$. Using this, constraints (5l) can be viewed as $s_i \geq \sum_{j:(j,i)\in\mathcal{A}} (s_j + t_{j,i}) x_{v^i,p^i-1,j} \ \forall\ i \in \mathcal{N}$. By constraint (5c), there is exactly one index j' such that $x_{v^i,p^i-1,j'} = 1$. Consequently, constraints (5l) enforce that the arrival time at each node i must be greater than or equal to the arrival time at the previously visited node plus the travel time. Similar reasoning can be done to derive constraints (5m): the only difference is that the arrival time to the depot depends on the vehicle v. Model (5) reads as a MBO problem, given the presence of both binary and continuous decision variables. The presence of constraints (5l) and (5m) makes the continuous relaxation of the problem nonconvex. As in the arc-based formulation, enforcing arrival times with an inequality rather than equality permits the possibility that a vehicle arrives early and waits.

Modeling the arrival times with continuous decision variables allows for an alternative formulation, with the aim of minimizing routes duration, without modifying the constraints set. This is achieved by switching objective (5a) with

$$\min_{\mathbf{x},\mathbf{s}} \sum_{v} s_v^d. \tag{6}$$

Routes duration is a typical metric to evaluate vehicle routes in the VRPTW.

III. SOLVING THE ROUTING PROBLEMS ON QUANTUM COMPUTERS

In this section, we discuss the reformulations needed to express the VRPTW formulations in a form suitable for quantum algorithms. In Sections III-A and III-B, most of the formulations are cast into (1), which is a standard form for a QUBO and enables the application of quantum algorithms such as VQE and QAOA. Due to the presence of continuous arrival times, the sequence-based formulation with continuous time, as presented in Section II-E, is an MBO and needs a different representation on quantum devices, which is discussed in Section III-C.

A. ROUTE- AND ARC-BASED FORMULATIONS AS QUBO

Both problems (2) and (3) have the common form

$$\min_{\mathbf{x}} \left\{ \mathbf{c}^\top \mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \in \{0,1\}^n \right\} \tag{7}$$

for some real vectors $\mathbf{b}$ and $\mathbf{c}$ and real matrix $\mathbf{A}$. The challenge is to construct a matrix $\mathbf{M}$ so that the QUBO standard form in (1) is equivalent to the binary linear program (7). Specifically, we would like our matrix to encode the quadratic penalty (or energy) function

$$H : \mathbf{x} \mapsto \mathbf{c}^\top \mathbf{x} + \rho \, \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$$

IEEE Transactions on Quantum Engineering

for a real constant $\rho > 0$, to be determined. This transformation is an exact penalty reformulation of (7), and it is consistent with the general suggestion for binary linear programs from [21, Sec. 3], as well as the transformation specific to the set partitioning problem from [21, Sec. 4.1].

The term $\|\mathbf{Ax} - \mathbf{b}\|^2$ expands to $\mathbf{x}^\top \mathbf{A}^\top \mathbf{Ax} - 2\mathbf{b}^\top \mathbf{Ax} + \mathbf{b}^\top \mathbf{b}$. Consequently, to obtain a QUBO, we set $\mathbf{M} = \rho \mathbf{A}^\top \mathbf{A} + \rho \mathbf{Diag}(-2\mathbf{A}^\top \mathbf{b}) + \mathbf{Diag}(\mathbf{c})$. The constant term $\|\mathbf{b}\|^2$ is then an added offset to match the original objective value.

The main challenge is finding the right value of ρ so that the minimization of H is equivalent to (7). The general reasoning is as follows. Looking at the data of the constraints of problems (2) and (3), and considering that the variables are binary, the smallest value that the penalty terms can take for an infeasible solution is ρ (when $\|\mathbf{Ax} - \mathbf{b}\|^2 = 1$). Thus, ρ needs to be big enough to overwhelm any decrease in the original objective by moving to an infeasible point. Imagine flipping each variable from 0 to 1 or *vice versa* depending on the sign of c_i; we can bound from above that change in objective by $\sum_i |c_i|$. Thus, $\rho > \sum_i |c_i|$ suffices. For the route-based formulation (2), this is established in more detail in Appendix B.

B. SEQUENCE-BASED FORMULATION AS QUBO

The sequence-based formulation (4) differs from the route- and arc-based formulations since it has bilinear equality constraints. To devise the QUBO reformulation, express the linear equality constraints (4b) and (4c) as $\mathbf{Ax} = \mathbf{b}$. Similarly, along with the bilinear equality constraints (4d) and (4e), these are turned into a penalty term

$$w : \mathbf{x} \mapsto \|\mathbf{Ax} - \mathbf{b}\|^2 + \sum_v \sum_{p<P} \sum_{(i,j)\notin\mathcal{A}} x_{v,p,i} x_{v,p+1,j} + \sum_v \sum_{p=2}^{P-1} \sum_{j:j\neq d} x_{v,p,d} x_{v,p+1,j}.$$

Consequently, an exact penalty reformulation of (4) is

$$\begin{aligned} \min_{\mathbf{x}} \ & \sum_v \sum_{p<P} \sum_{(i,j)\in\mathcal{A}} c_{i,j} x_{v,p,i} x_{v,p+1,j} + \rho w(\mathbf{x}) \\ \text{s.t. } & (4f), (4g) \\ & (4h), (4i) \\ & (4j), (4k) \\ & x_{v,p,i} \in \{0,1\} \quad \forall (v,p,i) \end{aligned} \tag{8}$$

where constraints (4f)–(4k) merely fix the values of certain variables.

For penalty parameter ρ sufficiently large, the solutions of (4) and (8) coincide. Looking at the specifics of the constraints and considering that the variables are binary, the smallest value that the penalty term $\rho w(x)$ can take for an infeasible solution is ρ (for $w(x) = 1$). As with the formulations in the previous subsection, ρ needs to be big enough to overwhelm any decrease in the original objective by moving to an infeasible point. This time, the change in objective when flipping each bilinear term depending on the sign of $c_{i,j}$ can be bounded by $\sum_v \sum_p \sum_{(i,j)\in\mathcal{A}} |c_{i,j}| = P|\mathcal{V}| \sum_{(i,j)\in\mathcal{A}} |c_{i,j}|$. Thus

$$\rho > P\,|\mathcal{V}| \sum_{(i,j)\in\mathcal{A}} |c_{i,j}|$$

suffices. As in Section III-A, let $\widetilde{\mathbf{M}} = \rho \mathbf{A}^\top \mathbf{A} + \rho \mathbf{Diag}(-2\mathbf{A}^\top \mathbf{b})$. We can add the other bilinear terms from the objective and penalty to get an objective in the form $\mathbf{x}^\top \mathbf{Mx}$, as desired for a QUBO.

C. VRPTW VIA ADMM-BASED HEURISTIC

In order to extend the range of mathematical formulations solvable on quantum near-term devices via variational-based approaches, Gambella and Simonetto [19] proposed a multi-block ADMM operator-splitting procedure. This iterative algorithm devises a decomposition for a specific class of MBOs into:

1) a QUBO subproblem to be solved by a QUBO solver oracle (or, on near-term quantum devices by quantum algorithms such as VQE or QAOA);
2) a convex constrained subproblem, which can be efficiently solved with classical optimization solvers [31].

The solutions obtained in each ADMM iteration are evaluated with a merit function, which evaluates the tradeoffs between feasibility and optimality. The authors devised conditions for the MBO to converge via ADMM to stationary points of a soft-constrained reformulation of the problem. In particular, the set of MBO constraints needs to have a continuous convex relaxation, which is not the case for the sequence-based formulation with continuous time. This motivates the adoption of a strategy to reformulate the continuous subproblem with convex approximations. In our case, the continuous subproblems are nonconvex, and they will be solved via a sequential convex approximation as follows.

Consider the cubic nonconvex constraints (5l)–(5m) of the sequence-based formulation with continuous times, namely:

$$s_i \geq \sum_{v\in\mathcal{V}} \sum_{p<P} \sum_{j:(j,i)\in\mathcal{A}} (s_j + t_{j,i}) x_{v,p,j} x_{v,p+1,i} \forall i \in \mathcal{N}$$

$$s_v^d \geq \sum_{p<P} \sum_{j:(j,d)\in\mathcal{A}} (s_j + t_{j,d}) x_{v,p,j} x_{v,p+1,d} \forall v \in \mathcal{V}.$$

The idea is to deal with them in the continuous problem of the ADMM framework by using *sequential convex programming* (see [32] and [33]). In particular, since we are splitting the problem onto the binary variables, we introduce continuous variable $u_{v,p,j} \in [0,1]$ for each three-indexed binary variable $x_{v,p,j}$, and set $u_{v,p,j} = x_{v,p,j}$. Then, in the continuous problem of the ADMM, where constraints (5l)–(5m) will become

$$s_i \geq \sum_{v\in\mathcal{V}} \sum_{p<P} \sum_{j:(j,i)\in\mathcal{A}} (s_j + t_{j,i}) u_{v,p,j} u_{v,p+1,i} \quad \forall i \in \mathcal{N}$$

$$s_v^d \geq \sum_{p<P} \sum_{j:(j,d)\in\mathcal{A}} (s_j + t_{j,d}) u_{v,p,j} u_{v,p+1,d} \quad \forall v \in \mathcal{V}.$$

such constraints can be compactly written as

$$\mathbf{g}(\mathbf{s}, \mathbf{u}) \leq 0. \tag{9}$$

Function $\mathbf{g}$ is not convex, since it is cubic in $(\mathbf{s}, \mathbf{u})$, but one can always use sequential convex programming approach to solve the continuous problem of the ADMM.

By using this strategy, the continuous problems of the ADMM converge to a local stationary point, and the overall ADMM strategy will remain a heuristic in general, but with the advantage that it limits the introduction of auxiliary binary decision variables in the QUBO subproblems and makes the solution of MBO on quantum devices a computationally tractable task.

IV. COMPARISON OF FORMULATIONS

We are now ready to showcase the different formulations on simulated quantum hardware and compare their numerical properties and performance.

First, in Section IV-A, we describe two examples that will serve as benchmarks to compare the mathematical formulations from a quantum computing perspective. We then report a summary of the qualitative modeling differences of the formulations in Section IV-B. Finally, quantitative comparisons are presented in Sections IV-C–IV-E and are meant to: 1) demonstrate the inherent difficulty in finding routes with good solution quality at a reasonable computational effort, using current classical algorithms; and 2) report the solution metrics obtained with the quantum state-of-the-art solution approaches on quantum devices.

A. EXAMPLE DEFINITIONS

1) MIRP WITH VARYING TIME HORIZON

We define an example inspired by the MIRP setting with a varying time horizon. The example is a modification of instance LR1_2_DR1_3_VC2_V6a in Group 1 of the MIRPLIB library [4]. This example involves two supply ports and three demand ports; the objective is to minimize travel costs while visiting each port frequently enough to remove or replenish its inventory. One characteristic of this example is that we can vary the time horizon of the problem; thus, we can effectively make the problem as large, in terms of number of nodes $|\mathcal{N}|$, as we want. See Appendix V for its data, interpretation as a VRPTW, and the specific steps required to obtain the various formulations from Section II.

2) SMALL VRPTW

We define a small three-customer example, originally from [30], with data reported in Fig. 1.

The instance has 11 valid routes, which define the feasibility set $\mathcal{R}$ for the route-based formulation (2): $(d, 1, d)$, $(d, 2, d)$, $(d, 3, d)$, $(d, 1, 2, d)$, $(d, 1, 3, d)$, $(d, 2, 1, d)$, $(d, 2, 3, d)$, $(d, 3, 1, d)$, $(d, 1, 2, 3, d)$, $(d, 2, 1, 3, d)$, $(d, 2, 3, 1, d)$. Routes $(d, 1, 2, 3, d)$ and $(d, 2, 3, 1, d)$ are optimal for the minimization of distance, with cost equal to 5. In order to define the arc-based formulation (3), we need to define the discrete time points $\mathcal{T}$. We use the starting and ending points of the time windows $\{0, 1, 2, 4, 7\}$. This happens to exclude the optimal route/sequence $(d, 2, 3, 1, d)$, which would require another time point $t \geq 8$ to allow the vehicle to return to the depot. For more complicated examples, it is often not viable to enumerate exhaustively the set of time periods to consider. For the sequence-based formulation, we make the assumption mentioned in Section II-D that vehicles arrive at the end of a node's time window. This means that we have to prune the set of arcs; consequently, the only valid arc from node 1 is $(1, d)$, and similarly, the only valid arc from node 3 is $(3, d)$.

B. QUALITATIVE COMPARISONS

Each of the VRPTW formulations considered in Section II has different strengths and weaknesses in terms of modeling, which are summarized in Table 1. Most of these characteristics have been discussed previously. For instance, the sequence-based formulation models timing discretely, since it effectively assumes that vehicles arrive at the end of a time window. We remark that in MIRPs, the constraints on vehicle capacity are not necessarily a strict requirement. This is because a demand node is often visited right after a supplier. Only the route-based formulation, as given, models the constraints on the vehicle capacities in full generality; the other formulations must be extended along the lines discussed in Sections II-C and II-D.

Another characteristic to consider is whether the formulations can handle inhomogeneous vehicles. The sequence-based and its continuous-time variant naturally can, since their variables are already indexed by vehicle v; transportation time or cost, for instance, could depend on the type of vehicle being used.

Finally, as mentioned in Section II-E, the total duration of the routes taken is a common alternative objective used in the VRPTW. The sequence-based formulation with continuous time can handle this, as can the route-based formulation by defining the cost of a route as the time that the vehicle arrives back at the depot. While the other formulations can take the arc costs $c_{i,j}$ to equal the arc travel time, this does not account for time spent waiting at a node.

C. PROBLEM SCALING

As a first step in comparing the different VRPTW formulations, we examine the resulting QUBO problems, as introduced earlier in Section III. The size and sparsity of the QUBO problem determine the resources needed to represent the problem on a quantum computer. Specifically, the number of logical qubits needed to represent the binary decision variables is equal to the dimension of the QUBO matrix $\mathbf{M}$, while the number of nonzero elements in the matrix has important implications for various solution methods. For example,

IEEE Transactions on Quantum Engineering

TABLE 1. Qualitative characteristics of the formulations

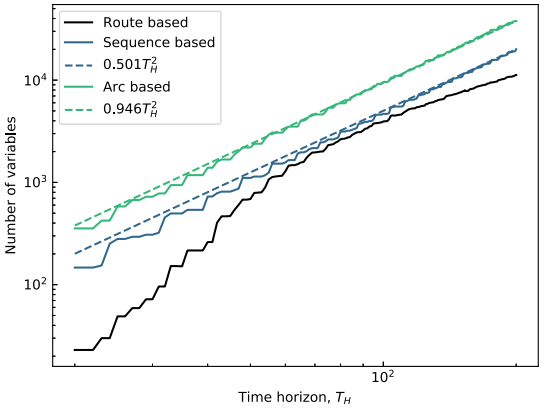
Characteristic	Route-based	Arc-based	Sequence-based	Sequence-based with continuous time
Models timing	Continuously	Discretely	Discretely	Continuously
Models capacity	Yes	No	No	No
Inhomogeneous vehicles	No	No	Yes	Yes
Routes duration objective	Yes	No	No	Yes

Optional features such as the possibility of handling inhomogeneous vehicles, and aiming for minimization of route duration are addressed.

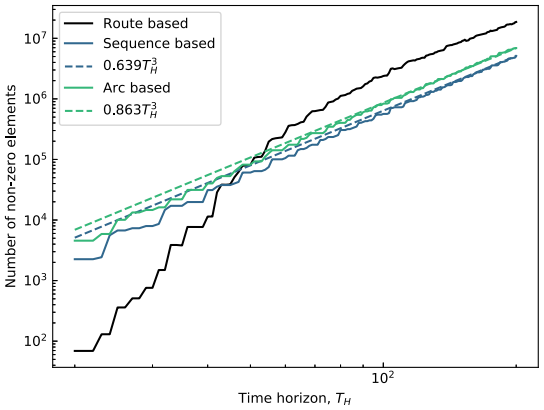
TABLE 2. QUBO problem size, connectivity, and nonzero elements for the MIRP example with varying time horizon

Formulation \ Time horizon		50	100	150	200
Route-based	size	687	3937	7608	11219
	connectivity	369	2198	4271	6746
	non-zeros	81191	2385791	8694599	18620459
Sequence-based	size	1106	4676	10647	20048
	connectivity	127	259	388	527
	non-zeros	60767	564641	1972880	5111778
Arc-based	size	2194	9584	21302	37858
	connectivity	143	379	559	773
	non-zeros	82041	835800	2848654	6901311

The statistics are reported for the route-based, sequence-based, arc-based formulations.



(a)



(b)

FIGURE 2. (a) Variables and (b) nonzero elements for the QUBO problems for the route-based, sequence-based, and arc-based formulations. It is clear that for the arc- and sequence-based formulations, the size (i.e., number of decision variables or qubits) increases as $O(T_H^2)$, while the nonzero elements in the QUBO matrix (e.g., nonzero correlations or Pauli strings) increases as $O(T_H^3)$.

in the case of QAOA, the quantum circuit contains a two-qubit gate for each off-diagonal nonzero entry in (an upper-triangular) $\mathbf{M}$. Another useful metric related to sparsity is the connectivity of the QUBO problem, defined as the degree of the graph with incidence matrix $\mathbf{M}$, and equivalent to the maximum number of nonzero entries per row (or column) of the matrix.

We now compare the formulations described in Section II, for the MIRP example with varying size introduced in Section IV-A1. As mentioned, we can increase the time horizon of the problem and subsequently obtain QUBOs of increasing size and complexity for each formulation.

Table 2 shows the growth with the time horizon of the size and the degree of connectivity of QUBOs for the route-based, sequence-based, and arc-based formulations. As discussed in Section II-E, the sequence-based formulation with continuous time is not directly expressed with a QUBO, but rather it is an MBO. However, the metrics for the discrete sequence-based formulation in Table 2 are highly informative for the effort required to solve MBO on quantum devices via ADMM. In particular, the QUBO subproblems of the ADMM heuristic for MBO have the same size of the sequence-based QUBO: this is because MBO and the sequence-based formulation (4) share the same combinatorial structure (i.e., same binary variables and constraints involving binary variables). The connectivity of the first QUBO subproblem of ADMM in MBO differs from that of the sequence-based QUBO by a constant term.

As expected, all formulations grow steadily with the time horizon, with the route-based formulation generating the smallest, but also least sparse, QUBO problems. However, recall that the route-based formulation depends on route generation heuristics. Similar heuristics could be applied to the sequence- and arc-based formulations as well. In general, classical variable reduction heuristics are critical when one is restricted by the number of qubits available. On the other hand, it seems to be intrinsically harder to directly control sparsity and connectivity, and thus, the arc- or sequence-based formulations may be preferable.

Fig. 2 shows a graphical representation of the growth of the size and density of the formulations as the time horizon increases. To emphasize the similarity between the sequence-based and arc-based, we also plot quadratic (i.e., $c_2 T_H^2$) and cubic (i.e., $c_3 T_H^3$) functions of the time horizon T_H. It is evident from the plots that the size of the corresponding QUBO increases as $O(T_H^2)$, and the number of nonzero elements in the QUBO matrix increases as $O(T_H^3)$. Thus, the two formulations are comparable, and while, in this particular case, the arc-based formulation has

approximately twice as many variables, in practice, one may want to consider other characteristics of the formulations (like how accurately timing needs to be modeled).

D. SOLUTION LANDSCAPE

The comparison between the different formulations can also be made in terms of the quality and quantity of feasible solutions. This is meant to describe the landscape of solutions on which the QUBO quantum solvers conduct their search for ground states. Variational algorithms, such as VQE and QAOA, are not guaranteed to converge to globally optimal solutions and might get stuck in locally optimal solutions [8]. Consequently, when using such an algorithm prone to converge to locally optimal solutions, a preferable formulation might be one where the local optimal solutions have small gaps relative to global optimality.[1]

In this section, we attempt to characterize the solution landscape for the route-, arc-, and sequence-based formulations of the MIRP example with varying sizes described in Section IV-A1 using the commercial branch-and-bound solver CPLEX [34]. In particular, we enumerate feasible solutions via CPLEX's populating routines and measure their quality via their relative difference with respect to the optimal solution. This is commonly known as optimality gap, defined as

$$\text{gap} = \frac{\text{sol} - \text{opt}}{|\text{opt}|}\%$$

where sol is the objective value of the solution and opt is the optimal solution value.

We consider the MIRP example with time horizons of 20 and 25. For each of these time horizons, each formulation achieves the same optimal objective value opt; these values are 2816.49 and 4457.15 for time horizons 20 and 25, respectively. This indicates a certain amount of equivalence between the formulations (for these specific instances, at least). Furthermore, the resulting problems are small enough to permit enumeration of all possible feasible solutions, for the most part. The exception is the arc-based formulation; even for these small instances, we are unable to compute all of its feasible solutions. The smallest arc-based formulation (with a time horizon of 20) has 355 variables and over 10 million feasible solutions, and enumerating the feasible solutions requires more than 100 GB of RAM and several hours of computation. The number of binary variables for these instances makes a complete enumeration impossible in a practical amount of time.

Continuing with just the route- and sequence-based formulations, we limit the characterization of the feasible solutions up to 50% optimality gap. In the unconstrained reformulation of the problem, a feasible solution beyond a certain optimality gap would be indistinguishable, in terms of the objective function, from infeasible solutions of the original problem.

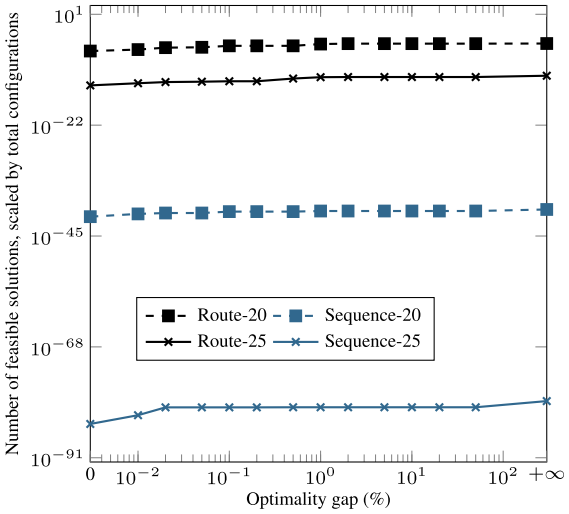


FIGURE 3. Cumulative number of feasible solutions (scaled by total number of configurations) within a certain percentage gap of the optimal solution for the route- and sequence-based formulations of the MIRP example with different time horizons. All optimal solutions are captured by an optimality gap equal to 0, while all feasible solutions are captured by an optimality gap equal to $+\infty$.

Fig. 3 shows the number of feasible solutions (scaled by the total number of configurations) within a certain percentage gap of the optimal solution for the route- and sequence-based formulations of the MIRP example. This figure essentially gives us the probability of randomly sampling (from the full configuration space) a feasible solution within a certain optimality tolerance. We immediately note that the size of the full configuration space (2^n where n is the number of variables) makes these probabilities dismally small, in general. Combining this with the quadratic growth in the number of variables for the sequence-based formulation observed in Section IV-C, this means that a modest increase in the length of the time horizon (20–25) reduces these probabilities by approximately 40 orders of magnitude.

Consequently, in Fig. 4, we instead scale by the total number of feasible solutions and plot the (cumulative) fraction of feasible solutions as a function of the optimality gap. Fig. 4 essentially gives us the performance of an algorithm that randomly samples only feasible solutions: for the route-based formulation with a time horizon of 25, such an algorithm would have a probability of 0.497 of producing a solution within 1% of optimal, versus a probability of 0.052 for the sequence-based formulation. Consequently, if we expect an algorithm (such as VQE or QAOA) to beat such random sampling, then this gives us a lower bound for its performance. Depending on the desired optimality gap, we are tempted to say that the route-based formulation is preferable, although this difference in the solution landscape between the formulations may have more to do with the preprocessing step of the route-based formulation, wherein a heuristic is used to

[1]We refer the reader to [19, Sec. 2] and references therein for a detailed discussion on VQE/QAOA possible quantum advantage w.r.t. classical solvers, especially for QUBOs. In this article, we note that our formulations are not dependent on a specific solver.

IEEE Transactions on Quantum Engineering

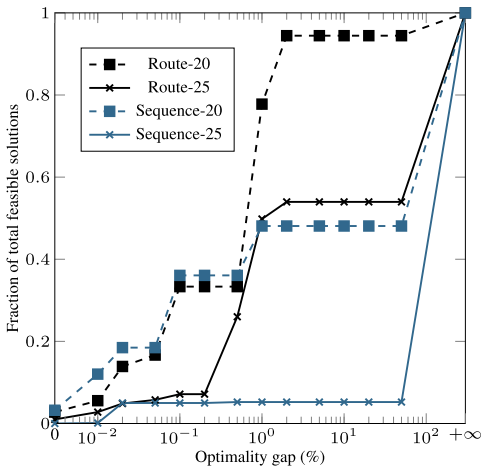


FIGURE 4. **Fraction of feasible solutions within a certain percentage gap of the optimal solution for the route- and sequence-based formulations of the MIRP example with different time horizons. The plot suggests that the route-based formulation tends to have a higher density of near-optimal feasible solutions.**

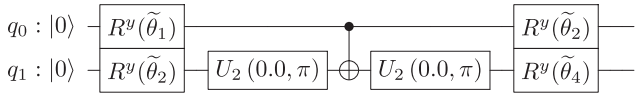


FIGURE 5. **Representation of the RY ansatz with two qubits and a depth of one.**

generate the set of routes. Determining the exact effect of the preprocessing step is an avenue for future research.

E. NUMERICAL EXPERIMENTS

In this section, we solve the small VRPTW example from Section IV-A2 using the IBM Quantum simulators and devices accessed through the open-source programming framework Qiskit [35], [36]. All experimental results are obtained using Qiskit's `QasmSimulator` backend. We focus the discussion on the sequence-based formulations since their size is small enough to analyze classically, yet large enough to provide some insights into the algorithm performance. This VRPTW instance requires 16 qubits to be represented as a QUBO.

1) SEQUENCE-BASED FORMULATION

For the discrete sequence-based formulation (4), we consider both the VQE with a standard hardware-efficient ansatz (RY) based on single-qubit rotations and two-qubit entangling gates [37] (see Fig. 5 for a representation of the variational form), and the QAOA, where the parameterization of the circuit is constructed by the alternate application of the cost Hamiltonian and a mixing operator [38], [39]. For the classical optimization part of the quantum algorithms, entailing the solution of optimal parameters for the quantum circuits, we consider two well-known gradient-free optimization algorithms: the simultaneous perturbation stochastic approximation (SPSA) optimizer [40], and constrained optimization by linear approximation (COBYLA) [41]. In the cases of both SPSA and COBYLA, the maximum number of function evaluations is set to 1000. We use the different ansatz/optimizer combinations (e.g., RY/SPSA, etc.), subsequently referred to as QUBO solvers, to directly solve the QUBO reformulation.

The sequence-based formulation for the small VRPTW example results in 16 decision variables. Out of the $2^{16} = 65\,536$ possible configurations or bitstrings, 2 are optimal, and another 2 are feasible (i.e., satisfy all constraints) but suboptimal. Since we are considering a depth one RY ansatz, this means that the classical optimization algorithms are optimizing a function of 32 real-valued parameters in the case of VQE. Meanwhile, we consider a $p = 1$ depth QAOA, and therefore, the optimization algorithms are optimizing a function of two real-valued parameters in the case of QAOA.

To compare the different QUBO solvers, we look at the probability of measuring an optimal configuration given the final optimized circuit. We will refer to this as the "single-shot" probability of success, p_s, which we determine by inspecting the probability amplitudes corresponding to the two known optimal configurations. However, since these solvers rely on classical optimization methods, there is a possibility of getting trapped at a local (sub) optimal set of circuit parameters. Consequently, to improve the robustness of the methods, they are typically run multiple times with different initial values for the parameters. We will do the same here, and sample each initial parameter value uniformly from $[0, 2\pi]$ ($[0, \pi]$ for the angles in the mixing part of QAOA). Thus, the performance of a QUBO solver is captured by its resulting distribution of single-shot probabilities of success. We will approximate this distribution by running each QUBO solver 250 times with these randomly sampled initial parameter values.

Table 3 reports various statistics for each solver's distribution of single-shot probabilities of success (as well as the probability of measuring any feasible solution, p_f, defined similarly). It is hard to draw practical conclusions from these statistics, however. We might suspect that RY/COBYLA is the solver of choice, since it has the highest average probability of success, and a reasonable fraction of the p_s are relatively large (greater than 10^{-3}). We could try to get a more holistic view of the distributions by plotting histograms, but we will instead propose the following statistic.

To put the performance of each QUBO solver in more practical applied terms, we consider the expected probability of success for each solver, as a function of the extra postoptimization evaluations (or "shots") we take of the final optimized quantum circuit. Given a "typical run" of a QUBO solver (essentially, a random sample p_s from the corresponding distribution), the probability of measuring an optimal configuration with N shots is $1 - (1 - p_s)^N$ (one minus the probability that an optimal configuration is *not* measured). Subsequently, we take the expected value of this under the distribution for p_s. Naturally, we approximate this using a sample-average estimate using our 250 samples of p_s for each solver.

The expected probabilities of success for the different solvers are presented in Fig. 6. Also included is the

TABLE 3. Statistics of single-shot probability of success and feasibility, for different solvers

Metric / Solver	Average p_f	Average p_s	Percentage of $p_s > 0.0001$	Percentage of $p_s > 0.001$
RY/COBYLA	0.345	0.158	17.2%	17.2%
RY/SPSA	0.045	0.018	29.6%	18.4%
QAOA/COBYLA	2.7×10^{-4}	1.6×10^{-4}	26.8%	3.2%
QAOA/SPSA	6.0×10^{-5}	3.2×10^{-5}	3.2%	0.0%

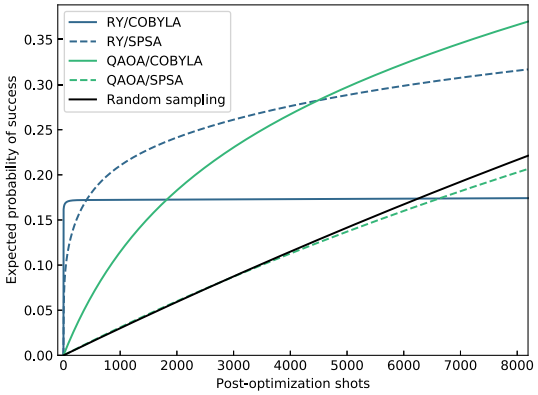


FIGURE 6. Expected probability of successfully measuring an optimal solution as a function of circuit evaluations or shots. These are the number of evaluations given an optimized circuit, that is, after the optimization phase.

corresponding function for uniform random sampling (in this case, p_s has a delta distribution, with mass at $\frac{2}{2^{16}}$). We include this as a benchmark; note that random sampling does not require any optimization effort. It is evident that the "best" choice of solver depends on the number of shots one is willing and able to take. For a low number of shots, the best performing solver is RY/COBYLA, which is quickly overtaken by RY/SPSA. With even more shots, QAOA/COBYLA performs even better. We speculate that this is because the QAOA ansatz can consistently produce a state with small, but nonzero, overlap with the optimal configurations. In contrast, an RY ansatz can exactly represent the optimal state (a basis state), but the tradeoff with this flexibility in the ansatz is that it is less forgiving to poor optimization of the circuit parameters.

2) SEQUENCE-BASED FORMULATION WITH CONTINUOUS TIME

The sequence-based formulation with continuous time needs 16 qubits for the QUBO subproblems to be solved via the ADMM heuristics. This is because the size of the QUBOs is the same as the QUBO reformulation of the discrete sequence-based model (4). The numerical results reported here are referred to the two-block implementation of ADMM with hyperparameters $\rho = 1001$ and $\beta = 1000$: this ensures that the ADMM terminates in a finite number of iterations, in case the continuous subproblem is convex (see [19] for a detailed discussion on the ADMM implementation and convergence properties). The QUBO subproblems of the ADMM heuristic are solved with VQE and QAOA as quantum solvers and COBYLA as a classical optimizer.

TABLE 4. Metrics obtained for the minimum-distance MBO via ADMM with the QUBO solvers QAOA and VQE

Solver	P_s	P_f	I	q_{opt}
QAOA	100.00%	100.00%	54	19.38%
VQE	100.00%	100.00%	32	10.42%

The continuous subproblems are solved via the sequential convex programming algorithm described in Section III-C. Preliminary computations conducted with COBYLA as a continuous ADMM solver showed that the linear approximations performed therein resulted in a considerable increase of the computational time.

Here, we have tested both the minimum-distance and minimum-time VRPTW formulations described in Section II-E. We have evaluated the MBO solutions via the following metrics:

1) probability of success, P_s, of ADMM. This is expressed as the percentage of ADMM runs that deliver an optimal solution for the problem. Note that the source of randomness of ADMM lies in the QUBO, which is solved via quantum algorithms;
2) probability of feasible solutions, P_f, found by ADMM, defined similarly to P_s;
3) number of iterations I for ADMM convergence;
4) percentage q_{opt} of QUBOs solved to optimality by the QUBO solver.

All metrics are obtained as average results over three runs of ADMM. The two QUBO solvers tested for ADMM are VQE and QAOA with COBYLA as an internal classical solver. Preliminary computations with the SPSA optimizer resulted in extremely slow ADMM convergence. For VQE, we chose the same ansatz tested for the QUBO formulations of Section IV-E1, since the underlying combinatorial structure of the optimization problem is very similar.

The minimum-distance formulation turns out to be very efficiently solved by the ADMM. As can be observed in Table 4, all three ADMM runs deliver a feasible and optimal route. Choosing VQE as QUBO solver results in a quicker convergence. Solving QUBOs in an exact fashion is not a guarantee for boosting ADMM convergence. Rather, a certain degree of inexactness in ADMM is beneficial for the overall solution quality [19], [42]. This point is particularly important since current computations on quantum devices are inevitably affected by errors and noise.

The minimum-time formulation is solved to optimality by the ADMM with VQE as a QUBO solver. For QAOA, Table 5 reports the metric obtained with two choices of circuit depth $p = 1, 3$. A larger depth for the variational form

IEEE Transactions on Quantum Engineering

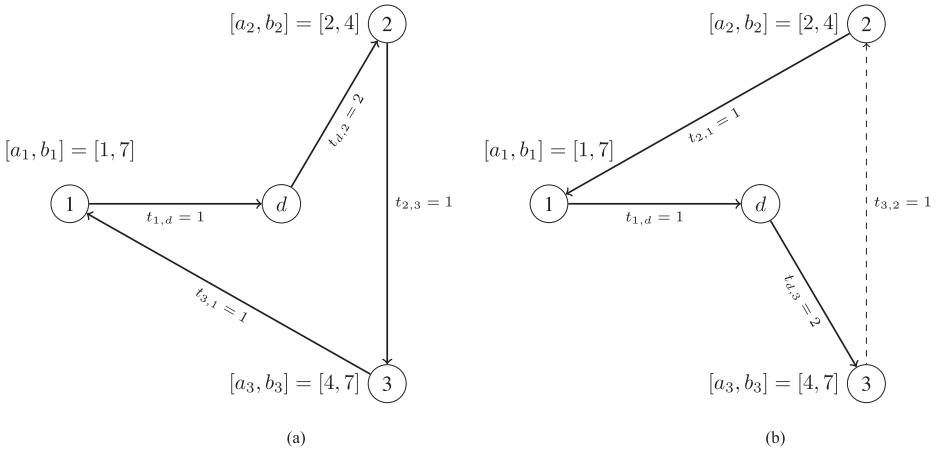


FIGURE 7. (a) and (b) Solutions found by the ADMM runs. Optimal solution for minimization of route duration is displayed on the left. The vehicle visits nodes 2 and 3 and waits one unit of time at node 3. At time 4, the vehicle leaves node 3 for visiting the remaining customer at node 1 and then reaches the depot at time 6. Infeasible solution for minimization of route duration is reported on the right. The vehicle visits node 3 and heads to node 2 at time 4. The infeasible arc for node 2 time windows is displayed with a dashed line. Finally, the vehicle visits the customer at node 1 and reaches the depot at time 7.

TABLE 5. Metrics obtained for the minimum-time MBO via ADMM with the QUBO solvers QAOA and VQE

Solver	P_s	P_f	I	q_{opt}
QAOA_1	33.33%	33.33%	50	18.87%
QAOA_3	33.33%	33.33%	44	26.46%
VQE	100.00%	100.00%	31	17.57%

QAOA has been tested with $p = 1$ ("QAOA_1"), and $p = 3$ ("QAOA_3").

helps ADMM converge in fewer iterations, and the success rates P_s and P_f of the ADMM with QAOA are $1/3$ in both cases. Specifically, in one simulation, the ADMM finds the feasible and optimal route with traveled time 6 displayed in Fig. 7(a). The other two ADMM runs with the QAOA QUBO solver deliver the route shown in Fig. 7(b), with infeasible arc connecting nodes 2 and 3. Future research work could explore the impact of choosing a different mixing Hamiltonian function in the QAOA algorithm, to aid the search for feasible routes.

For both VQE and QAOA as QUBO solvers, the ADMM exhibits convergence in a finite number of iterations, thanks to the sequential convex programming solver.

V. CONCLUSION

Size, sparsity, connectivity, model faithfulness, and difficulty to find optimal, or even feasible, solutions are all characteristics of a vehicle routing problem that depend heavily on the specific instances of interest and their mathematical formulation. Here, we have provided insights into these characteristics for the VRPTW. In particular, we have investigated these characteristics for four mathematical formulations amenable to being solved on current quantum hardware. Motivated by the MIRP, we have assessed metrics that indicate hardware requirements of tens of thousands of logical qubits to solve real-world business problems. As indicated in [6], this approximate threshold is also where it can be difficult to obtain good-quality solutions with reasonable computational effort on modern classical hardware.

We are tempted to ask the question: "Which formulation is best?" Naturally, there are tradeoffs, but each offers lessons to be learned. Since the number of qubits available on hardware will be limited in the near term, preprocessing heuristics, like those used in the route-based formulation, are needed to reduce the size of the problem. The route generation heuristics help make the route-based formulation the most compact one. Good preprocessing also seems to influence the higher density of near-optimal solutions of the route-based formulation, observed in Fig. 4. However, the route-based formulation has unfavorable sparsity and connectivity characteristics, which are also important metrics to consider for near-term hardware. This suggests using the route generation heuristics to reduce the size of the arc- or sequence-based formulations. It would be interesting to see if this helps reshape the solution landscape for these formulations as well.

The "best" QUBO solver depends on the number of samples one is willing to take from the optimized circuit. VQE and QAOA represent two extremes of the tradeoff between having a flexible ansatz that can represent the optimal solution and having a small enough number of parameters to facilitate the convergence of the classical optimizer to a high-quality solution. In our experiments, with a large enough number of samples, the QAOA/COBYLA solver had the highest probability of sampling an optimal configuration. Meanwhile, when the number of samples was limited, using VQE yielded a larger success probability.

For the sequence-based formulation with continuous time, the simulations with the sequential convex programming solver have shown the practical convergence of the ADMM heuristic to solutions with a large probability to be feasible and optimal. Furthermore, it has been shown that solving the QUBO subproblems on quantum devices at proven optimality is not a necessary condition to ensure the quality of the ADMM solutions. A certain degree of inexactness is tolerated by the ADMM algorithm, and this is a promising

feature to handle the inherent noise affecting the quantum algorithms on real devices.

Future work could extend the sequence-based with continuous time formulation to handle the vehicle capacity constraints and evaluate the quality of the ADMM solutions obtained.

APPENDIX A
CONSTRUCTION OF THE MIRP EXAMPLE

The example MIRP introduced in Section IV-A1 is a modification of instance LR1_2_DR1_3_VC2_V6a in Group 1 of the MIRPLIB library [4]. Here, we describe its data, interpretation as a VRPTW, and any specific steps required to obtain the various formulations from Section II.

A. DESCRIPTION OF THE EXAMPLE MIRP

In this example, we have two supply ports S1 and S2 and three demand ports D1, D2, and D3. Each supply port produces a good but has some limited amount of storage space for it. Similarly, each demand port consumes this good and has limited inventory of it. Different ports have different port fees as well. The port data for this example are reported in Table 6. The port fees have units of, e.g., dollars, while initial inventory and capacity have units of volume, and production and consumption rate have units of volume per time. The distances between the ports are given in Table 7.

A fleet of vessels is available; these vessels all have the same capacity of $Q = 300$ (volume units) and speed of 665 (distance per time, e.g., km/day). The travel cost per unit distance for these vessels is 0.09 (e.g., dollars per kilometer). We assume that the time horizon of interest begins at $t = 0$. We allow the end of the time horizon T_H to vary.

B. INTERPRETATION AS A VRPTW

The MIRP from Appendix V-A makes no mention of time windows or demand levels. In order to put this problem in the formalism of a VRPTW, we need to convert each port into a series of nodes with a fixed demand level and time window. To do this, we make the assumption that the vessels fully load at a supply port, travel to a demand port, and fully unload before returning to a supply port again. This can be a restrictive assumption for some applications; however, in this problem, each port has enough storage capacity to load or unload a full vessel, and therefore, it is unnecessary to consider partial unloading of a vessel. As mentioned in Section II, this assumption is reasonable in some situations like liquefied natural gas shipping.

To begin specifying the nodes, consider a supply port j. Its initial level of inventory (at $t = 0$) is I_j^0; at time t, the inventory is $I_j(t) = I_j^0 + t \cdot I_j^\Delta$. At time t^a when $I_j(t^a) = Q$, there is enough inventory to fully load a vessel. At time t^b when $I_j(t^b) = I_j^C$, the port runs out of storage space. These times define the first time window, during which the port must be visited. Now, assuming that the port has been visited p times already, the inventory level at time t is given by $I_j^0 + t \cdot I_j^\Delta - p \cdot Q$. The first time that the inventory reaches level Q again defines the beginning of the $(p+1)$th time window. The time at which the inventory reaches the capacity I_j^C defines the end of the $(p+1)^{th}$ time window. The result is that we define nodes indexed by port and number of previous visits; node (j, p) corresponding to supply port j that has been visited p times already has demand level $q_{(j,p)} = Q$ and time window $[a_{(j,p)}, b_{(j,p)}]$, where

$$a_{(j,p)} = (Q + pQ - I_j^0)/I_j^\Delta$$
$$b_{(j,p)} = (I_j^C + pQ - I_j^0)/I_j^\Delta.$$

For demand ports, the definitions are similar; the inventory level at time t after the port has been visited p times already is $I_j^0 + t \cdot I_j^\Delta + p \cdot Q$. The beginning of the time windows is defined by the times at which the inventory level reaches $I_j^C - Q$, at which point there is enough space to accept a full vessel to unload. The end of the time windows is defined by the times at which the inventory level reaches zero. Then, node (j, p) corresponding to demand port j that has been visited p times already has demand level $q_{(j,p)} = -Q$ and time window $[a_{(j,p)}, b_{(j,p)}]$, where

$$a_{(j,p)} = (I_j^C - Q - pQ - I_j^0)/I_j^\Delta$$
$$b_{(j,p)} = (0 - pQ - I_j^0)/I_j^\Delta.$$

Note that we must have $I_j^C \geq Q$ or else the time window is nonsensical ($a_{(j,p)} > b_{(j,p)}$), and when $I_j^C = Q$ (as is the case for port D3), the time window is degenerate.

For a given time horizon, we construct nodes for each port until the time windows are no longer a subset of the time horizon; that is, all nodes have $b_{(j,p)} \leq T_H$.

To finish specifying the VRPTW, we need the arc data. We define arcs between any supply node and any demand node, and *vice versa*. This enforces the assumption that vessels do not travel directly between supply ports or directly between demand ports. The travel time for an arc is simply the distance between the corresponding ports (see Table 7) divided by the vessel speed (665). The cost of the arc is the distance between the ports times the cost per unit distance (0.09), plus the port fee of the destination port (see Table 6). The original instance LR1_2_DR1_3_VC2_V6a includes time for loading and unloading vessels at the ports. For simplicity, we ignore this feature, although we could

TABLE 6. Port data for MIRP example

Port	S1	S2	D1	D2	D3
Initial Inventory, I_j^0	220	270	221	215	175
Storage Capacity, I_j^C	376	420	374	403	300
Production rate, I_j^Δ	47	42	-34	-31	-25
Port fee	30	85	60	82	94

TABLE 7. Distances between ports for MIRP example (distance are symmetric)

	S1	S2	D1	D2	D3
S1	0	212.34	5305.34	5484.21	5459.31
S2		0	5496.06	5674.36	5655.55
D1			0	181.69	380.30
D2				0	386.66
D3					0

IEEE Transactions on Quantum Engineering

include it by modifying the travel times by adding in the loading/unloading time at the destination port. Note this might make the travel times asymmetric.

Not all of the arcs defined this way are physically reasonable due to the timing. We remove arcs $((j, p), (j,' p'))$ where $a_{(j,p)} + t_{((j,p),(j,'p'))} > b_{(j,'p')}$; that is, the beginning of the time window of the origin node plus the travel time is greater than the end of the time window of the destination node.

The depot node in this problem is a dummy node, serving as an artificial source and sink for the vessels. Consequently, we assume that the initial loading Q_0 of the vessels is zero. In general, the number of vessels, their initial positions, and initial state (empty or full) are controlled through the specification of the entry arcs, which have the depot as their origin. The procedure for a general MIRP is as follows. For every vessel v, we add a dummy node dum_v with time window $[0, +\infty]$. If the vessel is initially empty, the dummy node has demand level $q_{\text{dum}_v} = 0$; an arc from the depot to this dummy node is added, and then, arcs from the dummy node to all supply nodes are added. If the vessel is initially fully loaded, the dummy node has demand level $q_{\text{dum}_v} = Q$ (to reflect the fact that this vessel visited a supply node sometime before the time horizon started); an arc from the depot to this dummy node is added, and then, arcs from the dummy node to all demand nodes are added. The travel times from the dummy node could reflect the geographic position of the vessel, for instance, that it is in the middle of the ocean. To finish specifying the network, we add exit arcs from any node (including the dummy ones) back to the depot at zero travel time and zero cost.

However, the original instance does not specify the initial states of the vessels, and therefore, we are less constrained in defining the entry arcs. Furthermore, because of how the arc- and sequence-based formulations handle the capacity constraints, adding these dummy nodes might not be necessary. Consequently, the handling of entry arcs is formulation-specific and discussed in the following subsection.

C. DETAILS FOR EACH FORMULATION

In this subsection, we go over any formulation-specific details required to handle the MIRP-as-VRPTW from the previous subsection. As mentioned, this includes the handling of entry arcs.

1) ROUTE-BASED FORMULATION

First, we specify the entry arcs for the formulation. Since vehicle capacity constraints are enforced through the definition of an allowed route, we must add dummy nodes, as discussed in Appendix V-B. We add entry arcs from the depot directly to any supply node with end of time window less than 14. Meanwhile, for every demand node with end of time window less than 14, we first add a dummy node with demand level Q. Then, we add arcs from the depot to this dummy node and then from the dummy node to the demand node (both with zero travel time and cost). This enforces the vessel to be fully loaded before it arrives at the demand node. The result is a set of seven vessels.

To finish specifying the route-based formulation, we need to define the set of routes $\mathcal{R}$. For this problem, we use a randomized greedy solution heuristic to propose routes. The core of this routine is given in Algorithm 1. This routine takes a scaling factor S and functions f_{time} and f_{node} that together assign a cost to visiting a node with a particular arrival time. The scaling factor S controls the amount of exploration/randomization that takes place, while f_{time} and f_{node} can be used to control the timing aspect of routes and which nodes the routes tend to favor visiting. Here, we set

$$f_{\text{time}} : t \mapsto \begin{cases} 0, & t \leq 10 \\ 100t, & t > 10 \end{cases}$$

$$f_{\text{node}} : i \mapsto \begin{cases} 0, & i \in \mathcal{N} \\ (T_H/6) \cdot 1500, & i = d \end{cases}.$$

This is intended to assign a high cost to routes that arrive at nodes later, thus promoting the generation of routes that greedily visit each node as early as possible, while simultaneously (through the action of f_{node}) discouraging early exit back to the depot.

The ultimate set of routes $\mathcal{R}$ is the unique set of routes generated by Algorithm 1 for increasing levels of randomization. Specifically, we run Algorithm 1 once for $S = 10^{-4}$, $\lfloor T_H \rfloor$ times for $S = 1$, and $\lfloor 10T_H \rfloor$ times for $S = +\infty$. For $S = +\infty$, the sampling at step 20 in Algorithm 1 is from a uniform distribution over the valid proposed nodes.

2) ARC-BASED FORMULATION

The additional data that we need to specify the arc-based formulation are the set of time periods $\mathcal{T}$. There are many options for determining this set; in general, we must balance detail and how finely we can model the timing of events, with how large the problem becomes. For this particular example, we choose the time periods to equal the set of integer time points that fall within any node's time window (besides the depot and any dummy node, which have infinite time windows), plus the initial time point $t = 0$. For this example, each node's time window contains at least one integer time point, and therefore, this gives us at least a little flexibility in the timing of the arrival of vessels at a node. Furthermore, this leads to some economy in the number of time points, since an integer time point might fall in the time window of multiple nodes.

Some care is required. Port D3 has degenerate time windows; however, they happen to fall at integer time points. Changing the initial inventory level so that it is not an integer multiple of the consumption rate would affect this.

The arc-based formulation does not explicitly enforce capacity constraints (in this example, vessel capacity is essentially enforced through the arcs, which only allow a vessel to visit an alternating sequence of supply and demand nodes). Consequently, the dummy nodes for enforcing the initial conditions of vessels are unnecessary. We add entry arcs from the depot directly to any supply node or demand node with end of time window less than 14. As in the route-based formulation, this results in seven vessels being available.

Algorithm 1: Randomized Greedy Solution Heuristic for Route Generation.

Require: $S \geq 0$, $f_{\text{time}} : [0, T_H] \to \mathbb{R}$, $f_{\text{node}} : \mathcal{N} \cup \{d\} \to \mathbb{R}$

Initialize proposed routes: $\mathcal{R}_P = \emptyset$
Initialize unvisited nodes: $\mathcal{N}_U = \mathcal{N}$
for $v \in \mathcal{V}$ **do**
  Initialize route and current arrival time: $i_1 = d$, $t_r = 0$.
  for $p = 2, 3, 4, \ldots$ **do**
    Construct costs for proposed next nodes:
    for $i \in \mathcal{N}_U \cup \{d\}$ **do**
      if $(i_{p-1}, i) \notin \mathcal{A}$ **then**
        continue
      end if
      $t_r^i = \max\{a_i, t_r + t_{i_{p-1},i}\}$
      if $t_r^i > b_i$ **then**
        continue
      else
        $f_i = c_{i_{p-1},i} + f_{\text{time}}(t_r^i) + f_{\text{node}}(i)$
      end if
    end for
    Sample next node according to softmax of negative costs:

$$i_p \sim \mathbb{P}(i|i_{p-1}) \equiv \frac{\exp(-f_i/S)}{\sum_j \exp(-f_j/S)}.$$

    Update route and arrival time: $r \leftarrow (d, i_2, i_3, \ldots, i_p)$, $t_r \leftarrow t_r^{i_p}$
    if $i_p = d$ **then**
      End building route; break iteration over p
    end if
  end for
  Update proposed routes: $\mathcal{R}_P \leftarrow \mathcal{R}_P \cup \{r\}$
  Update set of unvisited nodes by removing those visited by route $r = (d, i_2, i_3, \ldots, i_{P-1}, d)$: $\mathcal{N}_U \leftarrow \mathcal{N}_U - \{i_2, i_3, \ldots, i_{P-1}\}$
end for
Return $\mathcal{R}_P$

3) SEQUENCE-BASED FORMULATION

To specify the sequence-based formulation, we need a maximum number of stops P for each vehicle. Given the length of the time horizon T_H, we estimate this based on the shortest travel time ($5305.34/665 \approx 8$). Thus, we set $P = \lfloor T_H/8 \rfloor + 2$.

As in the arc-based formulation, dummy nodes for enforcing the initial conditions of vessels are unnecessary. We add entry arcs from the depot directly to any supply node or demand node with end of time window less than 14.

The sequence-based formulation does not directly enforce timing constraints. As mentioned, we enforce them in a conservative way by pruning the arc set: for any arc (besides an entry arc from the depot), if the *end* of the time window of the origin node plus the travel time is greater than the end of the time window of the destination node, then that arc is removed. We essentially enforce timing constraints by assuming that vessels always arrive at the end of a node's time window. Something similar could be done by assuming that vessels always arrive at the beginning of a node's time window.

APPENDIX B
PROOF OF SUFFICIENT PENALTY VALUE

In this appendix, we establish in detail a value of the penalty parameter that is sufficient to make the route-based formulation (2) equivalent to minimizing the energy function

$$H : \mathbf{x} \mapsto \sum_r c_r x_r + \rho \sum_i (1 - \textstyle\sum_r \delta_{i,r} x_r)^2.$$

Proposition 1: Assume that ρ satisfies

$$\rho > \sum_r |c_r| .$$

Then, $\mathbf{x}^*$ is a solution of $\min\{H(\mathbf{x}) : \mathbf{x} \in \{0,1\}^n\}$ and problem (2) is feasible, if and only if $\mathbf{x}^*$ solves problem (2) (where $n = |\mathcal{R}|$).

Proof: If $\mathbf{x}^*$ solves (2), then it is feasible; therefore, the penalty term is zero: $\sum_i (1 - \sum_r \delta_{i,r} x_r^*)^2 = 0$. Assume for a contradiction that there is an $\mathbf{x}^\dagger \in \{0,1\}^n$ with $H(\mathbf{x}^\dagger) < H(\mathbf{x}^*)$, or

$$\sum_r c_r x_r^\dagger + \rho \sum_i (1 - \textstyle\sum_r \delta_{i,r} x_r^\dagger)^2 < \sum_r c_r x_r^*. \tag{10}$$

If $\mathbf{x}^\dagger$ is feasible in (2), then the penalty term is zero, and therefore, $\sum_r c_r x_r^\dagger < \sum_r c_r x_r^*$, which contradicts the optimality of $\mathbf{x}^*$; thus, we must have that $\mathbf{x}^\dagger$ is infeasible in (2). Since $x_r^\dagger$ and $\delta_{i,r}$ are $\{0,1\}$-valued for all r and i, the smallest value that $\rho \sum_i (1 - \sum_r \delta_{i,r} x_r^\dagger)^2$ can take is ρ (since it is infeasible, it cannot be zero). Meanwhile, by the (generalization of) the Cauchy–Schwarz inequality, $-\sum_r c_r (x_r^\dagger - x_r^*) \leq \|\mathbf{c}\|_* \|\mathbf{x}^\dagger - \mathbf{x}^*\|$ for any norm $\|\cdot\|$ and its dual norm $\|\cdot\|_*$. In particular, using the infinity norm, we have $-\sum_r c_r (x_r^\dagger - x_r^*) \leq \|\mathbf{c}\|_1 \cdot 1$. Using $\rho \leq \rho \sum_i (1 - \sum_r \delta_{i,r} x_r^\dagger)^2$ and $-\|\mathbf{c}\|_1 \leq \sum_r c_r (x_r^\dagger - x_r^*)$ and plugging into (10), we have

$$-\|\mathbf{c}\|_1 + \rho < 0$$

but upon rearranging and using the definition of the one-norm, we see this contradicts the assumption that $\rho > \sum_r |c_r|$. Thus, $\mathbf{x}^* \in \arg\min_{\mathbf{x}} H(\mathbf{x})$.

Conversely, assume that $\mathbf{x}^\dagger$ solves $\min_{\mathbf{x}} H(\mathbf{x})$, and that problem (2) is feasible. We have $\min_{\mathbf{x}} H(\mathbf{x})$ must be less than or equal to the minimum objective value of (2); $H(\mathbf{x})$ equals the objective of (2) on the feasible set of (2), and the minimization of H is over a superset of the feasible set of (2), so the minimum must be less. Thus, we just need to establish that $\mathbf{x}^\dagger$ is feasible for (2). So, assume for a contradiction that $\mathbf{x}^\dagger$ is not feasible. By assumption, there exists $\mathbf{x}^*$ feasible in (2). Since $\mathbf{x}^\dagger$ minimizes H, we have

$$\sum_r c_r x_r^\dagger + \rho \sum_i (1 - \textstyle\sum_r \delta_{i,r} x_r^\dagger)^2 \leq \sum_r c_r x_r^*.$$

We can proceed exactly as before to obtain $-\|\mathbf{c}\|_1 + \rho \leq 0$, which still contradicts the assumption that $\rho > \sum_r |c_r|$. Therefore $\mathbf{x}^\dagger$ is feasible in (2) and, thus, optimal. ■

ACKNOWLEDGMENT

The authors would like to thank Jakub Marecek for help in developing this work and Laurent White and Gilad Ben-Shach for constructive feedback on the manuscript. IBM and Qiskit are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.

REFERENCES

[1] P. Toth and D. Vigo, *Vehicle Routing: Problems, Methods, and Applications*, 2nd ed. Philadelphia, PA, USA: SIAM, 2014, doi: 10.1137/1.9781611973594.

[2] R. Baldacci, A. Mingozzi, and R. Roberti, "Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints," *Eur. J. Oper. Res.*, vol. 218, no. 1, pp. 1–6, 2012, doi: 10.1016/j.ejor.2011.07.037.

[3] L. A. Wolsey and G. L. Nemhauser, *Integer and Combinatorial Optimization*. Hoboken, NJ, USA: Wiley, 1999, doi: 10.1002/9781118627372.

[4] D. J. Papageorgiou, G. L. Nemhauser, J. Sokol, M.-S. Cheon, and A. B. Keha, "MIRPLib—A library of maritime inventory routing problem instances: Survey, core model, and benchmark results," *Eur. J. Oper. Res.*, vol. 235, no. 2, pp. 350–366, 2014, doi: 10.1016/j.ejor.2013.12.013.

[5] United Nations, "Review of maritime transport," United Nations, New York, NY, USA, Tech. Rep. UNCTAD/RMT/2018, 2018.

[6] D. J. Papageorgiou, M.-S. Cheon, S. Harwood, F. Trespalacios, and G. L. Nemhauser, "Recent progress using matheuristics for strategic maritime inventory routing," in *Modeling, Computing and Data Handling Methodologies for Maritime Transportation*. Berlin, Germany: Springer, 2018, pp. 59–94, doi: 10.1007/978-3-319-61801-2_3.

[7] P. K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, and S. Woerner, "Improving variational quantum optimization using CVaR," *Quantum*, vol. 4, 2020, Art. no. 256, doi: 10.22331/q-2020-04-20-256.

[8] G. Nannicini, "Performance of hybrid quantum-classical variational heuristics for combinatorial optimization," *Phys. Rev. E*, vol. 99, no. 1, 2019, Art. no. 013304, doi: 10.1103/PhysRevE.99.013304.

[9] A. V. Fiacco and G. P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, vol. 4. Philadelphia, PA, USA: SIAM, 1990, doi: 10.1002/zamm.19720520716.

[10] C.-Y. Wang and D. Li, "Unified theory of augmented Lagrangian methods for constrained global optimization," *J. Global Optim.*, vol. 44, no. 3, 2009, Art. no. 433, doi: 10.1007/s10898-008-9347-1.

[11] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014, *arXiv:1411.4028*.

[12] A. Peruzzo *et al.*, "A variational eigenvalue solver on a photonic quantum processor," *Nat. Commun.*, vol. 5, 2014, Art. no. 4213, doi: 10.1038/ncomms5213.

[13] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *Phys. Rev. X*, vol. 10, no. 2, Jun. 2020, Art. no. 021067, doi: 10.1103/PhysRevX.10.021067.

[14] E. Farhi and A. Harrow, "Quantum supremacy through the quantum approximate optimization algorithm," 2016, *arXiv:1602.07674*.

[15] G. E. Crooks, "Performance of the quantum approximate optimization algorithm on the maximum cut problem," 2018, *arXiv:1811.08419*.

[16] A. Gilliam, S. Woerner, and C. Gonciulea, "Grover adaptive search for constrained polynomial binary optimization," 2019, *arXiv: 1912.04088*.

[17] F. L. Brandão, R. Kueng, and D. Stilck França, "Faster Quantum and classical SDP approximations for quadratic binary optimization," 2019, *arXiv: 1909.04613*.

[18] L. Braine, D. J. Egger, J. Glick, and S. Woerner, "Quantum algorithms for mixed binary optimization applied to transaction settlement," 2019, *arXiv:1910.05788*.

[19] C. Gambella and A. Simonetto, "Multiblock ADMM heuristics for mixed-binary optimization on classical and quantum computers," *IEEE Trans. Quantum Eng.*, vol. 1, 2020, Art. no. 3102022, doi: 10.1109/TQE.2020.3033139.

[20] B. Heim, E. W. Brown, D. Wecker, and M. Troyer, "Designing adiabatic quantum optimization: A case study for the traveling salesman problem," 2017, *arXiv:1702.06248*.

[21] A. Lucas, "Ising formulations of many NP problems," *Frontiers Phys.*, vol. 2, pp. 1–14, 2014, doi: 10.3389/fphy.2014.00005.

[22] S. Feld *et al.*, "A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer," *Frontiers ICT*, vol. 6, 2019, Art. no. 13, doi: 10.3389/fict.2019.00013.

[23] H. Irie, G. Wongpaisarnsin, M. Terabe, A. Miki, and S. Taguchi, "Quantum annealing of vehicle routing problem with time, state and capacity," in *Proc. Int. Workshop Quantum Technol. Optim. Problems*, 2019, pp. 145–156, doi: 10.1007/978-3-030-14082-3_13.

[24] B. Kallehauge, "Formulations and exact algorithms for the vehicle routing problem with time windows," *Comput. Oper. Res.*, vol. 35, no. 7, pp. 2307–2330, 2008, doi: 10.1016/j.cor.2006.11.006.

[25] B. Sutton, K. Y. Camsari, B. Behin-Aein, and S. Datta, "Intrinsic optimization using stochastic nanomagnets," *Sci. Rep.*, vol. 7, 2017, Art. no. 44370, doi: 10.1038/srep44370.

[26] X. Yin, B. Sedighi, M. Varga, M. Ercsey-Ravasz, Z. Toroczkai, and X. S. Hu, "Efficient analog circuits for boolean satisfiability," *IEEE Tran. Very Large Scale Integr. Syst.*, vol. 26, no. 1, pp. 155–167, Jan. 2018, doi: 10.1109/TVLSI.2017.2754192.

[27] P. L. McMahon *et al.*, "A fully programmable 100-spin coherent Ising machine with all-to-all connections," *Science*, vol. 354, no. 6312, pp. 614–617, 2016, doi: 10.1126/science.aah5178.

[28] A. Douglass, A. D. King, and J. Raymond, "Constructing SAT filters with a quantum annealer," in *Theory and Applications of Satisfiability Testing—SAT 2015*, M. Heule and S. Weaver, Eds. Cham, Switzerland: Springer, 2015, pp. 104–120, doi: 10.1007/978-3-319-24318-4_9.

[29] S. Tsukamoto, M. Takatsu, S. Matsubara, and H. Tamura, "An accelerator architecture for combinatorial optimization problems," *FUJITSU Sci. Technol. J.*, vol. 53, pp. 8–13, 2017. [Online]. Available: https://www.fujitsu.com/global/about/resources/publications/fstj/archives/vol53-5.html

[30] M. Desrochers, J. Desrosiers, and M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," *Oper. Res.*, vol. 40, no. 2, pp. 342–354, 1992, doi: 10.1287/opre.40.2.342.

[31] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004, doi: 10.1017/CBO9780511804441.

[32] J. Duchi, "Sequential convex programming," *Lecture Notes EE364b*, Stanford Univ., Stanford, CA, USA, 2018.

[33] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods.*, (ser. MOS-SIAM Series on Optimization). Philadelphia, PA, USA: SIAM, 2000, doi: 10.1137/1.9780898719857.

[34] *IBM ILOG CPLEX Version 12.10*, IBM, Armonk, NY, USA, 2020.

[35] H. Abraham *et al.*, "Qiskit: An Open-Source Framework for Quantum Computing," 2019, doi: 10.5281/zenodo.2562110.

[36] Qiskit: An Open-Source Quantum Computing Software Development Framework, 2016. [Online]. Available: https://qiskit.org/

[37] A. Kandala *et al.*, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, pp. 242–246, 2017, doi: 10.1038/nature23879.

[38] 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*, vol. 12, no. 2, 2019, Art. no. 34, doi: 10.3390/a12020034.

[39] Z. Wang, N. C. Rubin, J. M. Dominy, and E. G. Rieffel, "*XY* mixers: Analytical and numerical results for the quantum alternating operator ansatz," *Phys. Rev. A*, vol. 101, Jan. 2020, Art. no. 012320, doi: 10.1103/PhysRevA.101.012320.

[40] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Trans. Autom. Control*, vol. 37, no. 3, pp. 332–341, Mar. 1992, doi: 10.1109/9.119632.

[41] S. Gomez and J. P. Hennart, *Advances in Optimization and Numerical Analysis*, vol. 275. Berlin, Germany: Springer, 2013, doi: 10.1007/978-94-015-8330-5.

[42] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *J. Sci. Comput.*, vol. 78, no. 1, pp. 29–63, 2019, doi: 10.1007/s10915-018-0757-z.