Solving the shipment rerouting problem with quantum optimization techniques

Yarkoni, S.; Huck, A.; Schülldorf, H.; Speitkamp, B.; Tabrizi, M.S.; Leib, M.; ... ; Lalla-Ruiz, E., Voss, S.

Solving the Shipment Rerouting Problem with Quantum Optimization Techniques

Sheir Yarkoni[1,2(✉)], Andreas Huck[3], Hanno Schülldorf[3], Benjamin Speitkamp[3], Marc Shakory Tabrizi[3], Martin Leib[1], Thomas Bäck[2], and Florian Neukart[1,2]

[1] Volkswagen Data:Lab, Munich, Germany
sheir.yarkoni@volkswagen.de
[2] LIACS, Leiden University, Leiden, The Netherlands
[3] Deutsche Bahn AG, Berlin, Germany

Abstract. In this work we develop methods to optimize an industrially-relevant logistics problem using quantum computing. We consider the scenario of partially filled trucks transporting shipments between a network of hubs. By selecting alternative routes for some shipment paths, we optimize the trade-off between merging partially filled trucks using fewer trucks in total and the increase in distance associated with shipment rerouting. The goal of the optimization is thus to minimize the total distance travelled for all trucks transporting shipments. The problem instances and techniques used to model the optimization are drawn from real-world data describing an existing shipment network in Europe. We show how to construct this optimization problem as a quadratic unconstrained binary optimization (QUBO) problem. We then solve these QUBOs using classical and hybrid quantum-classical algorithms, and explore the viability of these algorithms for this logistics problem.

1 Introduction

Quantum computing has garnered increased interest in recent years in both research and industrial settings. This novel technology holds the promise of solving computationally intractable problems asymptotically faster than their classical counterparts in a variety of application areas [1–3]. The public availability of quantum devices from commercial entities such as D-Wave Systems, Google, and IBM have produced a variety of results showcasing novel algorithms and potential use-cases for quantum computing in fields such as quantum machine learning [4,5], logistics/scheduling [6,7], quantum chemistry [8,9], and more [10]. Of particular interest is the potential of quantum processing units (QPUs) to affect the field of optimization, making the technology attractive to both research and industry experts. Currently, variational quantum optimization techniques are the main targets of promising research, and hold the highest potential of gaining advantage using quantum processors. For more details about the different paradigms of quantum computing and their mathematical backgrounds we refer the reader to [11–13].

M. Mes et al. (Eds.): ICCL 2021, LNCS 13004, pp. 502–517, 2021.
https://doi.org/10.1007/978-3-030-87672-2_33

Optimization problems for quantum algorithms and other similar heuristics are typically formulated as either Ising Hamiltonians (posed in a $\{-1, 1\}$ basis) or quadratic unconstrained binary optimization (QUBO) problems (in a $\{0, 1\}$ basis). Finding the minimum of an Ising Hamiltonian, or its equivalent QUBO, is known to be an NP-hard problem in the worst case [14], meaning many difficult and well-known optimization problems have such representations [15]. For our work, we focus on the QUBO formulation of optimization problems:

$$\text{Obj}(Q, \mathbf{b}) = \mathbf{b} \cdot Q \cdot \mathbf{b}^T. \tag{1}$$

Here, Q is an $N \times N$ matrix representing interaction terms between variables in the binary vector $\mathbf{b}$ with N variables. Therefore, the first step in using quantum optimization algorithms is finding a valid QUBO representation of the problem to be solved. In this paper we focus on designing a QUBO representation for an industrially motivated logistics optimization problem, and attempt to optimize the QUBOs using both hybrid quantum-classical and purely classical QUBO solvers. Additionally, we consider a representation by a mixed integer program (MIP) which we optimize by the standard solver Gurobi.

We investigate a problem motivated by an application in logistics: the less-than-truckload network service design. Less-than-truckload (LTL) denotes shipments not exceeding a maximum weight significantly below a full truck load. The transport of a single shipment follows the sequence of (a) a collecting truck run, followed by (b) one or several linehaul truck runs (including handling of the shipment) and ending with (c) a distributing truck run to the shipment's final destination. Our work focuses on the design of the linehaul network, step (b). The linehaul network for LTL is made up by the set of terminals and timetable based truck runs, connecting the terminals and thereby producing the long-hauls of all the shipments entering the network. Taking limitations on transport times into account, the forwarding of the shipments shall be as cost efficient as possible. One key factor for cost efficiency is the consolidation of multiple shipments in jointly utilized trucks, at least regarding parts of their individual linehaul paths through the network. This measure targets an increase of truck utilization. However, the consolidation of multiple shipments with different origins and destinations in jointly utilized trucks requires detours of shipments. As detours come at a cost, the network design searches for an optimal trade-off between detour costs and the benefit of increased truck utilization. We focus on this central trade-off decision and call the reduced problem the *shipment rerouting problem* (SRP). We provide an illustrative example with two shipments in Fig. 1.

The input to the SRP includes a set of terminals, their distances to each other, and the numbers of available trucks connecting the terminals. Moreover, we have a set of shipments, each with a set of possible routes of intermediate terminals. These possible routes already comply with constraints like maximum transport time or maximum detour factor. They include the direct route from the origin to the destination of the shipment, which are the default for all shipments. Other candidate routes for rerouting are constructed in a pre-processing step based on the graphical structure of the terminals and the distances between them.

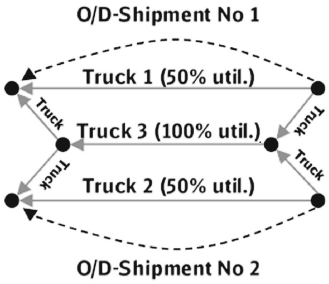

Fig. 1. An example of the SRP with two shipments. The default routing (Trucks 1 and 2 carrying their respective shipments at 50% capacity each) is optimized by replacing Trucks 1 and 2 with a single truck (Truck 3) which can be fully utilized. The cost of rerouting each shipment to the route serviced by Truck 3 is offset by the removal of Trucks 1 and 2, thereby reducing the overall distance travelled to deliver the shipments.

Thus, a subset of shipments may be rerouted through alternate routes in order to reduce the overall distance all trucks travel to deliver the shipments. Each shipment has a size (volume, weight, etc.), and each truck has a corresponding capacity, i.e. an upper bound for the total shipment size that can be loaded. For our purposes, we denote the shipment sizes and truck capacities with respect to volume, and refer to them as such throughout the rest of this work. We note that our mathematical formulations equally admit other quantities.

A shipment cannot be split across different routes. However, for transporting a shipment between two terminals, we may split it to distribute it on multiple trucks (this is necessary especially for shipments with a large volume). Given the input, the task is to decide on a route for each of the given shipments, which may include overlaps between shipments. Consequently, the result includes the number of required trucks in the network and which terminals are connected by truck runs in which frequency.

The rest of this paper is organized as follows: Sect. 2 discusses previous literature relevant to this analysis, and Sect. 3 motivates the QUBO construction for this problem based on a MIP representation and details the specifics of the QUBO construction used throughout this work. Section 4 outlines the input data, solvers, and experimental design of our analysis. Section 5 presents the results from those experiments, and Sect. 6 summarizes the conclusions derived from our work.

2 Previous Works

In [16], Ding et al. solve a network design problem with a quantum annealing approach. However, this problem is different as it searches for the best terminal locations while the arc costs are linear (a hub location problem). In [6] it is shown how to form QUBO representation of a simple traffic flow combinatorial optimization problem. In that work, individual vehicles are given multiple candidate

routes whose intersection needs to be minimized. This route-generation procedure is used in our work, but with opposite intent, our objective is to consolidate as many routes as possible.

Other examples of logistics and scheduling applications in quantum computing include flight-path conflict resolution [17] and railway train rescheduling [18]. Both examples use elements from a generic job-shop scheduling formulation for quantum annealing [7]. While these applications are qualitatively similar to some aspects of our work, the shipment rerouting problem is unique for quantum annealing as not only the selection of routes for each shipment is variable, but also the number of trucks used on each edge along the path is selected by the optimization. Typically, for example in job-shop scheduling, the number of machines and jobs are inputs to the QUBO construction. Our formulation thus incorporates elements from both scheduling (route selection in [6] and [18]) and packing problems (canonical problems in NP [15]).

Because of its cost structure, the SRP problem is closely related to the fixed-charge multi-commodity network design (FCMND) problem which has not been investigated in the field of quantum computing. However, it has been studied extensively in the past. Exact algorithms are usually based on branch-and-cut approaches and Bender's decomposition– see [19] for an overview. A particular problem for exact algorithms is that the lower bound is hard to improve. There are also heuristic approaches to solve this problem, using evolutionary algorithms [20] and simulated annealing [21], among others.

3 Constructing MIP and QUBO Representations

The MIP representation of the SRP is straightforward to formulate, as we can use multiple kinds of variables (binary, integer, real) and constrain the solutions explicitly. Therefore, we start with a MIP representation which we then transform to a QUBO.

3.1 Constructing the MIP Representation

We assume that the connectivity of the terminals can be represented as a weighted directed graph G where the vertices V are the terminals and the edges E between them represent the ability to transport shipments from any single terminal to another; in other words, we have an edge $e \in E$ from a terminal a to a terminal b if there are trucks available driving from a to b. These trucks are called the *trucks on* e and their number is denoted by $t_{\max}(e)$. The weight of e is the distance from a to b and is denoted by $d(e)$. For each shipment s, $v(s)$ denotes its volume and $R(s)$ denotes the set of all routes that can be used to transport s (*candidate routes* of s). For each edge e, $R(e)$ denotes the set of all candidate routes containing e. A shipment s is *scheduled* on some edge e if s is transported using an associated candidate route r containing e, i.e. $r \in R(s) \cap R(e)$.

In our scenarios, all trucks have the same volume capacity, denoted by c_{vol}. Moreover, all shipments have different origin-destination pairs so that no two

different shipments have common candidate routes (however, their candidate routes may overlap). Therefore, for each candidate route r, we have a unique shipment $s(r)$ that can be transported using r.

We want to transport each shipment on an associated candidate route such that the total distance of all used trucks in the network is minimized. To represent this problem by a MIP, we introduce a binary decision variable y_r for each candidate route r that is 1 if r is used to transport $s(r)$, and 0 otherwise. For each edge e, we introduce a non-negative integer variable t_e with maximal value $t_{\max}(e)$ representing the number of used trucks on e. We represent the problem by the following MIP:

Objective: Minimize the total truck distance

$$\sum_{e \in E} d(e) \cdot t_e \tag{2}$$

with respect to the following constraints:
Route-shipment constraints: For each shipment s, exactly one associated candidate route is used, i.e.

$$\sum_{r \in R(s)} y_r = 1. \tag{3}$$

Capacity constraints: For each edge e, the total volume of all shipments scheduled on e does not exceed the total volume capacity of the used trucks on e, i.e.

$$\sum_{r \in R(e)} v(s(r)) \cdot y_r \leq c_{\text{vol}} \cdot t_e. \tag{4}$$

The capacity constraints ensure that on each edge e, enough trucks are used to transport all shipments scheduled on e because we can split shipments to optimally exploit the truck capacities. Note that in an optimal solution, each truck number t_e is as small as possible, namely $\left\lceil \sum_{r \in R(e)} v(s(r)) \cdot y_r / c_{\text{vol}} \right\rceil$. In that case, for each edge e, we can completely fill all used trucks on e except possibly one truck that is partially filled.

3.2 Constructing the QUBO Representation

Contrary to a MIP, a QUBO only contains binary variables and an objective function to be minimized without explicit constraints. However, the quadratic summands arising from Eq. (1) allow us to include penalty terms that emulate the MIP constraints.

Our QUBO formulation uses the binary variables y_r for the candidate routes r. In replacement of the integer variables t_e for the edges e, we use modified binary representations of their values in the QUBO based on a concept in [15]: for each edge e, we define $T(e)$ to be the set of all powers of two less than or equal $t_{\max}(e)$, and for each $n \in T(e)$, we introduce a binary variable $t_{e,n}$ in

order to represent the number of used trucks on e by $\sum_{n\in T(e)} n\cdot t_{e,n}$. In this way, we can represent at least each number up to $t_{\max}(e)$, i.e. each allowed truck number. However, the maximal representable number is $2n_{\max}-1$ where $n_{\max}$ is the maximal value in $T(e)$. Therefore, to avoid representations of numbers greater than $t_{\max}(e)$, we reduce the coefficient $n_{\max}$ in $\sum_{n\in T(e)} n\cdot t_{e,n}$ by the surplus $s := 2n_{\max}-1-t_{\max}(e)$. The new expression is denoted by

$$\sum_{n\in T(e)} \overline{n}\cdot t_{e,n}, \tag{5}$$

i.e. we have $\overline{n_{\max}} = n_{\max}-s = 1+t_{\max}(e)-n_{\max}$ and $\overline{n}=n$ for each $n\neq n_{\max}$. Now we can still represent each number up to $t_{\max}(e)$ but no other numbers. In our QUBO, we reformulate the total truck distance (2) as

$$\sum_{e\in E} d(e)\cdot \sum_{n\in T(e)} \overline{n}\cdot t_{e,n}. \tag{6}$$

To encode the route-shipment constraints (3), note that they are linear equalities of the form $A=B$ where only binary variables occur. Each such constraint is implemented in our QUBO by adding the summand $M\cdot(A-B)^2$ where M is a large penalty factor ensuring that the constraint is fulfilled at least in all optimal solutions of our QUBO. We will later discuss how to define a suitable penalty factor.

The capacity constraints (4) cannot be implemented in that way (after reformulation using the representations (5)) because they are inequalities of the form $A\leq B$. However, such a constraint can be transformed into an equality $A+\ell=B$ by using a non-negative slack variable ℓ. Note that the slack of the capacity constraint for each edge e is the wasted volume in the used trucks on e (*volume capacity slack* on e). Unfortunately, contrary to the numbers of used trucks, these slacks might be large or fractional values so that their representations might require many binary variables, making our QUBO intractable.

To overcome this problem, we discretize the shipment volumes into *bins*: We virtually divide the loading area of each truck into the same number c_{bin} of equally sized bins. c_{bin} is called the *bin capacity* of the trucks. Each bin can only be used for transporting one shipment and has the volume capacity $c_{\text{vol}}/c_{\text{bin}}$. Hence, for each shipment s, the number $b(s)$ of bins needed to transport s is given by $b(s)=\lceil v(s)\cdot c_{\text{bin}}/c_{\text{vol}}\rceil$. Instead of the volume capacity slacks, we now have to represent the *bin capacity slack* on each edge e, i.e. the number of unused bins in the used trucks on e. These slacks are more tractable because they are integers that can be assumed to be less than c_{bin} (we will see this later).

On the other hand, if c_{bin} is too small, then the bin volume capacity $c_{\text{vol}}/c_{\text{bin}}$ is large so that we may obtain several partially filled bins in the trucks, especially if shipments exist that are smaller than the bin volume capacity (recall that we cannot use a bin for transporting more than one shipment). Hence, we may not optimally exploit the truck capacities any more which may increase the number of used trucks. We can improve the situation by multiplying the bin capacity,

i.e. by subdividing each bin into the same number of smaller bins.[1] Therefore, c_{bin} is a crucial parameter for the QUBO construction: more bins may lead to a better exploitation of the truck capacities, but at the cost of larger bin capacity slacks to be represented. In our experiments, we used the bin capacity 10, which was an empirically-determined compromise.

For each edge e, we introduce a non-negative integer variable ℓ_e representing the bin capacity slack on e. Then we obtain a new *discretized* MIP by modifying the capacity constraints (4) as follows:

Capacity Constraints: For each edge e, we have

$$\sum_{r \in R(e)} b(s(r)) \cdot y_r + \ell_e = c_{\text{bin}} \cdot t_e. \tag{7}$$

These constraints imply the former ones (because $v(s) \leq b(s) \cdot c_{\text{vol}}/c_{\text{bin}}$ for each shipment s) and may even be stronger (due to a worse exploitation of the truck capacities). Similar to the former MIP, in an optimal solution of the new MIP, each truck number t_e is as small as possible, namely $\left\lceil \sum_{r \in R(e)} b(s(r)) \cdot y_r / c_{\text{bin}} \right\rceil$. Therefore, each bin capacity slack ℓ_e is less than c_{bin} so that we can represent these values in the QUBO as follows: we define L to be the set of all powers of two less than c_{bin}, and for each edge e and for each $m \in L$, we introduce a binary variable $\ell_{e,m}$ such that the bin capacity slack of e is

$$\sum_{m \in L} m \cdot \ell_{e,m}. \tag{8}$$

In this way, we can represent at least each number less than c_{bin}, i.e. each relevant bin capacity slack (it doesn't matter if we can represent further numbers).

Using the representations (5) and (8), we can reformulate the capacity constraints (7) as follows:

Capacity Constraints: For each edge e, we have

$$\sum_{r \in R(e)} b(s(r)) \cdot y_r + \sum_{m \in L} m \cdot \ell_{e,m} = c_{\text{bin}} \cdot \sum_{n \in T(e)} \overline{n} \cdot t_{e,n}. \tag{9}$$

Similar to the route-shipment constraints, these capacity constraints are implemented in the standard way by summands of the form $M \cdot (A - B)^2$. Putting all components together, we obtain the following formulation of the QUBO:

[1] Simply increasing the bin capacity may worsen the situation. For instance, suppose that $v(s) = c_{\text{vol}}/2$ for each shipment s so that $b(s) = \lceil c_{\text{bin}}/2 \rceil$. If $c_{\text{bin}} = 2$, then $b(s) = 1$ so that we can put two shipments into a truck. But if $c_{\text{bin}} = 3$, then $b(s) = 2$ so that we can put only one shipment into a truck.

$$\sum_{e\in E} d(e)\cdot \sum_{n\in T(e)} \overline{n}\cdot t_{e,n} + M\cdot \sum_{s\in S}\left(\sum_{r\in R(s)} y_r - 1\right)^2$$
$$+M\cdot \sum_{e\in E}\left(\sum_{r\in R(e)} b(s(r))\cdot y_r + \sum_{m\in L} m\cdot \ell_{e,m} - c_{\text{bin}}\cdot \sum_{n\in T(e)} \overline{n}\cdot t_{e,n}\right)^2. \quad (10)$$

Here, all variables are as before, and S is the set of all shipments in the problem.

We now choose the penalty factor M to ensure that only feasible solutions are present in the global optimum of the QUBO objective, so that it is never energetically favorable to violate one of the constraints in favor of minimizing the total track distance. In general, we may choose any M greater than the total truck distance $d(\mathit{feas})$ of any known feasible solution *feas* (for instance, the solution transporting each shipment on its direct route). To see the correctness of M, consider an optimal solution *opt* and suppose that *opt* violates a constraint. Then the *opt*-value of the QUBO objective is at least M and thus greater than $d(\mathit{feas})$. But since *feas* is feasible, $d(\mathit{feas})$ is also the *feas*-value of the QUBO objective, contradicting the optimality of *opt*.

The QUBO requires many more variables than the MIP in Sect. 3.1. For each truck number variable t_e in the MIP, we have $|T(e)| = \lceil \log_2(t_{\max}(e)+1)\rceil$ variables $t_{e,n}$ to represent its values. Additionally, we have $|L|\cdot|E| = \lceil \log_2 c_{\text{bin}}\rceil \cdot |E|$ variables $\ell_{e,m}$ to represent the bin capacity slacks.

3.3 Improvements to the QUBO

To make our QUBO more tractable to a QUBO solver, we now construct improvements which do not necessarily reduce the QUBO size but the range of the coefficients for implementing certain capacity constraints. We define the *potential shipments* on an edge e to be the shipments with a candidate route containing e (i.e. these shipments can be scheduled on e) and we denote by $S(e)$ the set of all these shipments. For each shipment set S, we define $t_{\text{vol}}(S)$ to be the minimal number of trucks that are sufficient to transport all shipments in S referring to the volume capacity, i.e. $t_{\text{vol}}(S) = \lceil \sum_{s\in S} v(s) \,/\, c_{\text{vol}}\rceil$.

Note that we must use at least one truck on an edge e if at least one shipment is scheduled on e. Moreover, if $t_{vol}(S(e)) = 1$, then that truck is sufficient to transport *all* that shipments. Therefore, we can replace the corresponding capacity constraints in (9) by simpler ones which do not require the bin discretization:

Simple Capacity Constraints : For each edge e with $t_{\text{vol}}(S(e)) = 1$ and for each $r \in R(e)$, we have

$$y_r \le t_{e,1}. \quad (11)$$

This constraint can be implemented by adding the summand $M\cdot(y_r - y_r\cdot t_{e,1})$ to our QUBO. Note that $y_r - y_r\cdot t_{e,1} = 0$ if $y_r \le t_{e,1}$, and that $y_r - y_r\cdot t_{e,1} = 1$ otherwise. Therefore, it is unfavorable to violate the constraint.

This concept can be generalized to a larger class of edges. To do that, we define the *shipment sets* of an edge e to be the subsets of $S(e)$, and we call a shipment set S *n-minimal* for some positive integer n if $t_{\text{vol}}(S) \geq n$ and $t_{\text{vol}}(S') < n$ for each proper subset S' of S. Moreover, S is called *minimal* if it is n-minimal for some n (n may be not unique if all shipments in S exceed the volume capacity). We generalize the concept of (11) to all edges that only have a few minimal shipment sets. These edges we define *good* and all other edges *bad*.

Note that the only 1-minimal shipment sets are the singletons $\{s\}$ with some shipment s. Therefore, if e is an edge with $t_{\text{vol}}(S(e)) = 1$ (as in (11)), then each minimal shipment set of e is 1-minimal and thus a singleton. Hence we may definitely define each such edge to be good. For our scenarios, we obtained the best results when defining an edge e to be good if $t_{\text{vol}}(S(e)) \leq 2$ or if there are at most 8 potential shipments on e. Then most edges in our scenarios are good, and for all other edges, the number of minimal shipment sets is usually such huge that applying our approach does not improve our QUBO.

We let E_{good} and E_{bad} denote the set of all good and bad edges, respectively. For each good edge e, we introduce new binary variables: the trucks on e are numbered by $1, ..., t_{\max}(e)$ and for each $n = 1, ..., t_{\max}(e)$, we have a binary decision variable $x_{e,n}$ that is 1 iff truck n on e is used. For technical reasons, we additionally define $x_{e,t_{\max}(e)+1}$ to be the constant 0. The total truck distance (6) is then reformulated as

$$\sum_{e \in E_{\text{bad}}} d(e) \cdot \sum_{n \in T(e)} \overline{n} \cdot t_{e,n} + \sum_{e \in E_{\text{good}}} d(e) \cdot \sum_{n=1}^{t_{\max}(e)} x_{e,n}. \tag{12}$$

As before, our QUBO contains implementations of all route-shipment constraints (3) and of the capacity constraints (9) for all bad edges. For the good edges, we have the following constraints which generalize the concept of (11):

Good Capacity Constraints : Let e be a good edge. Then for each n-minimal shipment set S of e with $n \leq t_{\max}(e) + 1$, we have

$$\prod_{s \in S} \sum_{r \in R(s,e)} y_r \leq x_{e,n} \tag{13}$$

where $R(s, e)$ denotes the set of all candidate routes of s containing e, i.e. $R(s, e) = R(s) \cap R(e)$. Particularly, if $n = t_{\max}(e)+1$, we have $\prod_{s \in S} \sum_{r \in R(s,e)} y_r = 0$.

Before showing how to implement these constraints, we first verify their correctness. We show that in a feasible solution of our QUBO, we have enough used trucks on each good edge e to transport all shipments scheduled on e. Let U denote the set of these shipments. For each $n = 1, 2, ..., t_{\text{vol}}(U)$, we choose a smallest possible subset S_n of U with $t_{\text{vol}}(S_n) \geq n$. Then each S_n is n-minimal and $\prod_{s \in S_n} \sum_{r \in R(s,e)} y_r$ is always 1 because each shipment s in S_n is scheduled on e so that $\sum_{r \in R(s,e)} y_r = 1$. Hence if $n \leq t_{\max}(e) + 1$, then by (13) applied to S_n, we

obtain $x_{e,n} = 1$. Now since $x_{e,t_{\max}(e)+1} = 0$, we see that $t_{\text{vol}}(U) \leq t_{\max}(e)$ and that truck n on e is used for each $n = 1, 2, ..., t_{\text{vol}}(U)$. These are enough trucks to transport all shipments in U. On the other hand, the constraints (13) do not force $x_{e,n} = 1$ for some $n > t_{\text{vol}}(U)$. To see that, consider an n-minimal shipment set S of e. Then since $t_{\text{vol}}(S) > t_{\text{vol}}(U)$, S contains a shipment s that is not in U, i.e. s is not scheduled on e. Hence $\sum_{r \in R(s,e)} y_r = 0$ and thus $\prod_{s \in S} \sum_{r \in R(s,e)} y_r = 0$.

To implement the constraints (13), we consider an n-minimal shipment set S of some good edge e with $n \leq t_{\max}(e) + 1$. Let $s_1, s_2, ..., s_k$ denote the shipments of S, and for each $i = 1, 2, ..., k$, let Σ_i denote the expression $\sum_{r \in R(s_i,e)} y_r$. Hence we have to implement the constraint $\Sigma_1 \cdot \Sigma_2 \cdot ... \cdot \Sigma_k \leq x_{e,n}$. We consider a solution fulfilling all route-shipment constraints (3) so that $\Sigma_i \leq 1$ for each $i = 1, 2, ..., k$. If $k = 1$, then similarly to (11), we implement the constraint $\Sigma_1 \leq x_{e,n}$ by the summand $M \cdot (\Sigma_1 - \Sigma_1 \cdot x_{e,n})$. Note that since $\Sigma_1 \leq 1$, we obtain $\Sigma_1 - \Sigma_1 \cdot x_{e,n} = 0$ if $\Sigma_1 \leq x_{e,n}$, and $\Sigma_1 - \Sigma_1 \cdot x_{e,n} = 1$ otherwise. Now assume $k \geq 2$. To emulate the product $\Sigma_1 \cdot \Sigma_2 \cdot ... \cdot \Sigma_k$, we introduce auxiliary binary variables $z_2, z_3, ..., z_k$. First, we implement the constraint $z_2 = \Sigma_1 \cdot \Sigma_2$ by the summand $\Omega_2 := M \cdot \big(\Sigma_1 \cdot \Sigma_2 + (3 - 2 \cdot \Sigma_1 - 2 \cdot \Sigma_2) \cdot z_2\big)$. Note that since $\Sigma_1 \leq 1$ and $\Sigma_2 \leq 1$, we obtain $\Omega_2 = 0$ if $z_2 = \Sigma_1 \cdot \Sigma_2$, and $\Omega_2 \in \{M, 3M\}$ otherwise. Now analogously, for each $i = 3, 4, ..., k$, we implement the constraint $z_i = z_{i-1} \cdot \Sigma_i$ by the summand $\Omega_i := M \cdot \big(z_{i-1} \cdot \Sigma_i + (3 - 2 \cdot z_{i-1} - 2 \cdot \Sigma_i) \cdot z_i\big)$. This forces $z_i = \Sigma_1 \cdot \Sigma_2 \cdot ... \cdot \Sigma_i$ for each $i = 2, 3, ..., k$. Finally, we implement the constraint $z_k \leq x_{e,n}$ by the summand $M \cdot (z_k - z_k \cdot x_{e,n})$.

Note that the summands Ω_i may be negative in a solution where $\Sigma_i \geq 2$ for some indices i. Therefore, it may be favorable (or at least not unfavorable) to violate some route-shipment constraints (3). To avoid such violations, we add the auxiliary summand $\Delta_i := \frac{M}{2} \cdot \Sigma_i \cdot (\Sigma_i - 1)$ to our QUBO for each $i = 1, 2, ..., k$ which is 0 if all route-shipment constraints are fulfilled. It is straightforward to verify that in all solutions, $\Omega_2 + \Delta_1 + \Delta_2 \geq 0$ and $\Omega_i + \Delta_i \geq 0$ for each $i = 3, 4, ..., k$ so that it is unfavorable now to violate any constraint.

4 Experiments and Data

The inputs used in this work were generated from a real-world network of hubs in Europe belonging to DB Schenker. The specific locations and distances between hubs have been abstracted to comply with data protection laws, but are representative of the real-world network. Connections between hubs correspond to serviced routes between hubs. We use one graphical model to represent the entire hub network, and generate multiple inputs based on different numbers of shipments: 30, 50, 80, and 100 shipments. In all inputs, every shipment s_{ij} travels from one hub (v_i) to another (v_j). The direct route, $v_i \rightarrow v_j$ along e_{ij}, is always the first candidate route for s_{ij}. The other candidate routes are generated by a staggered *k-shortest path* approach: shipments are categorized by their OD-distance, and for each category the k shortest paths are calculated where k

increases with respect to the OD-distance of the category. For example, shipments up to 200 km have one alternative route while shipments over 1000 km have up to 10 routes. The volume of the shipments is randomly generated using an adapted exponential distribution, resulting in many smaller shipments and few larger shipments.

In this study we use multiple solvers for our SRP instances and gauge the viability of QUBOs as representations of the problem. We provide a brief introduction and motivation for each solver.

Direct Shipments. We consider the "direct shipment" solution to the SRP as a simple baseline. The direct solution is computed by routing every shipment (s_{ij}) along its most direct path (e_{ij}). Since every shipment origin/destination is unique in our instances, this equates to using one truck per edge for every shipment.

Simulated Annealing. Simulated annealing is a well-known heuristic optimization algorithm for combinatorial optimization [22]. The algorithm involves probabilistically flipping individual variables' states proportionally to the objective value change such a flip would induce and the current "temperature" of the system. Candidate solutions are initialized at random, and the temperature parameter is initialized to infinity; solutions are slowly "cooled" and the temperature is lowered until the solutions settle in local optima and variable flips no longer occur. Simulated annealing has been used extensively in benchmarking studies related to quantum computing [23,24]. The specific implementation of simulated annealing in this analysis was from the Python package dimod [25].

Tabu Search. This algorithm is another metaheuristic for combinatorial optimization, operating on the principle that searching already-discovered solutions should be actively discouraged (a "tabu list"). Individual variables' states are flipped based on their likelihood of importance in the global optimum [26]. Solutions which worsen the objective function value may be explored by the search if no other variable flip is possible, which allows for both global and local refinement of solutions. The Python package used for Tabu can be found here [25].

Gurobi. Optimal solutions and optimality bounds were produced by solving the MIP in Sect. 3.1 using Gurobi, an exact branch-and-bound solver. The benefit of using Gurobi is that a bound on the optimality of the solutions is provided. Given that the objective function units are the same for all solvers, this optimality gap can be used for all solvers in this analysis. The run-time allocated to Gurobi was 24 h per input to obtain good bounds for each instance.

D-Wave Hybrid Solver. The smallest instance in our test set required 787 QUBO variables. While small for the application, this is larger than could be

solved on D-Wave QPUs at the time of experiments. Instead, a proprietary hybrid classical-quantum algorithm offered by D-Wave Systems was used, called the Hybrid Solver Service (HSS), which has been used in previous applications [27], and admits QUBOs with up to 10k binary variables. The HSS uses a QPU to optimize clusters of variables, allowing one to leverage the use of a quantum processor without the overhead of embedding. However, this hybrid algorithm does not allow direct access to control the QPU in its inner loop. Therefore, we consider the HSS as a black-box optimizer, and measure the performance as a function of the timeout parameter, similar to Gurobi and other proprietary solvers.

5 Results

The consolidated results appear in Fig. 2. While the total run-time of Gurobi was 24 h to obtain good lower bounds, good solutions with an optimality gap of less than 10 percent were already found after a few minutes for all instances. For the 30 and 50 shipment instances, we also obtained provably optimal solutions within the first few minutes of optimization. The solutions from Gurobi were significantly better than those obtained by solving the QUBO formulation. However, this is possibly due to both the fact that Gurobi is an exact solver and the way in which the MIP is discretized, as explained in Sect. 3.2. Tabu search was able to find a near-optimal solution for the 30 shipment instance, but was unable to find even feasible solutions for any of the other instances. Simulated annealing was able to find feasible solutions, but only in the largest case of 100 shipments was the solution better than the direct shipment approach. The D-Wave HSS was able to find better-than-direct solutions for the 30, 50, and 80 shipment instances. To attempt a fair comparison, each QUBO solver was given roughly the same amount of time per test instance. However, the specific parameter choices corresponding to such times were found and set by hand. We include the parameter settings chosen in Appendix A.

Table 1. Number of variables and terms needed to describe the problem instances using binary encoding and good capacity constraints.

Shipments	Routes	QUBO variables	QUBO terms
30	223	787	4856
50	428	1526	16315
80	752	2305	40594
100	925	3318	59014

Throughout our initial experiments, we found that increasing the number of possible routes for each shipment does not directly correlate with improved solutions to the original problem (lower total truck km). This is due to the fact

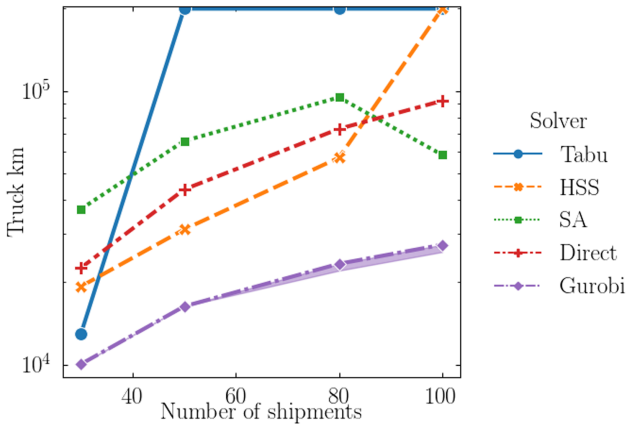

Fig. 2. Performance of all solvers used in the experiments. We display the results in units of truck kilometers for ease of comparison. Simulated annealing (SA), Tabu, and the D-Wave HSS are QUBO solvers, Gurobi is a MIP solver, and the direct solutions are the simple baseline of one truck per shipment (explained in Sect. 4).

that each additional route creates more minima and a more rugged landscape. It is important to note that given the way we construct the QUBO– no trucks along an edge is a valid solution– increasing the number of possible routes can only create additional minima, not remove minima that have already been created. Given this insight, it became even more important to consider the number of QUBO terms (shown in Table 1) and to improve the QUBO as outlined in Sect. 3.3.

6 Conclusions

In this work we motivated a logistics optimization problem based on a real-world use-case, the shipment rerouting problem. This problem models the distance minimization of a simple objective function– total number of truck kilometers used to send shipments between nodes in a graph. We presented methods to translate this problem to a QUBO form using both simple minimization objectives (truck kilometers as weights on decision variables), and hard constraints (knapsack-like constraints on edges in the graph) to test both quantum and classical optimization algorithms. We further presented methods to optimize the QUBO representation in attempts to improve the performance of the algorithms used in our experiments. We found that there was significant amount of work in finding such valid QUBO representations. Despite the relatively straightforward description of the problem, to correctly model the solution landscape was more subtle, and required multiple iterations of derivations, as explained throughout the text. Nonetheless, we found it an informative exercise, as the lessons learned can be applied to future work.

Of the algorithms tested, Gurobi performed the best despite being an exact branch-and-bound algorithm. Of the heuristics, we found that the D-Wave HSS was able to find better than greedy solutions for the smaller problem sizes tested. We stress that given our small test bed we cannot conclude any one solver being the best relative to the others, nor was this the intention. Furthermore, we find that the bar we define as "acceptable" (finding solutions that are better than direct shipments) was surprisingly difficult for the heuristics to beat. This is important to note since simulated annealing was able to find valid solutions for all the problem sizes, but better-than-direct for only the largest problem. From this, together with the long run-times required to find these valid solutions, we conclude that this type of logistics optimization problem may not benefit from transformation to a QUBO for the purpose of being solved with heuristics. The growth in the number of variables required to solve such relatively small problems was a bottleneck that could not be compensated for. However, with the advent of error-corrected quantum processors in the future, it is possible that this bottleneck can be overcome. Until then, our future research will be dedicated to finding real-world optimization problems that are better-suited for current quantum technologies.

A Solver Parameters

Here we present the time allocated to each solver in Table 2, and the corresponding parameters in Table 3. For the D-Wave HSS, we limit the 30 and 50 shipment instances to only 5 minutes of run-time. We note that these 5 minutes were sufficient for the problems tested. Because we could not control the usage of the QPU in the D-Wave HSS, we report the QPU run-time in the timing results rather than a parameter. All software solvers were executed using single-threaded programs.

Table 2. Table of run-time allocated to each solver in the experimental setup.

Instance	Simulated annealing	Tabu	HSS
30	1 h	1 h	5 min (QPU: 3.0 s)
50	1 h	1 h	5 min (QPU: 1.4 s)
80	1 h	1 h	1 h (QPU: 3.61 s)
100	1 h	1 h	1 h (QPU: 4.34 s)

Table 3. Parameter sets used for each solver. Parameters not mentioned were set to default values.

Instance	Simulated annealing	Tabu	HSS
30	2500 samples, 50000 sweeps	1 h timeout	5 min timeout, $use\_qpu = True$
50	1600 samples, 50000 sweeps	1 h timeout	5 min timeout, $use\_qpu = True$
80	1000 samples, 50000 sweeps	1 h timeout	1 hr timeout, $use\_qpu = True$
100	500 samples, 50000 sweeps	1 h timeout	1 hr timeout, $use\_qpu = True$

References

1. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 124–134 (1994)
2. Deutsch, D., Jozsa, R.: Rapid solution of problems by quantum computation. Proc. Roy. Soc. Lond. Ser. A Math. Phys. Sci. **439**(1907), 553–558 (1992)
3. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96, pp. 212–219. Association for Computing Machinery, New York (1996)
4. Amin, M.H., Andriyash, E., Rolfe, J., Kulchytskyy, B., Melko, R.: Quantum Boltzmann machine. Phys. Rev. X **8**(2), 021050 (2018)
5. Alexander, C., Shi, L., Akhmametyeva, S.: Using quantum mechanics to cluster time series. arXiv:1805.01711 (2018)
6. Neukart, F., Compostella, G., Seidel, C., von Dollen, D., Yarkoni, S., Parney, B.: Traffic flow optimization using a quantum annealer. Front. ICT **4**, 29 (2017)
7. Venturelli, D., JJ Marchand, D., Rojo, G.: Quantum annealing implementation of job-shop scheduling. arXiv:1506.08479 (2015)
8. Streif, M., Neukart, F., Leib, M.: Solving quantum chemistry problems with a D-wave quantum annealer. In: Feld, S., Linnhoff-Popien, C. (eds.) QTOP 2019. LNCS, vol. 11413, pp. 111–122. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-14082-3_10
9. Quantum, G.A.I.: Hartree-fock on a superconducting qubit quantum computer. Science **369**(6507), 1084–1089 (2020)
10. Venturelli, D., Kondratyev, A.: Reverse quantum annealing approach to portfolio optimization problems. Quant. Mach. Intell **1**(1), 17–30 (2019). https://doi.org/10.1007/s42484-019-00001-w
11. Johnson, M.W., et al.: Quantum annealing with manufactured spins. Nature **473**(7346), 194–198 (2011)
12. Farhi, E., Goldstone, J., Gutmann, S.: A quantum approximate optimization algorithm. arXiv:1411.4028 (2014)
13. Aharonov, D., Van Dam, W., Kempe, J., Landau, Z., Lloyd, S., Regev, O.: Adiabatic quantum computation is equivalent to standard quantum computation. SIAM Rev. **50**(4), 755–787 (2008)
14. Barahona, F.: On the computational complexity of ising spin glass models. J. Phys. A Math. Gener. **15**(10), 3241 (1982)
15. Lucas, A.: Ising formulations of many np problems. Front. Phys. **2**, 5 (2014)
16. Ding, Y., Chen, X., Lamata, L., Solano, E., Sanz, M.: Implementation of a hybrid classical-quantum annealing algorithm for logistic network design. SN Comput. Sci. **2**(2), 68 (2021)

17. Stollenwerk, T., et al.: Quantum annealing applied to de-conflicting optimal trajectories for air traffic management. IEEE Trans. Intell. Transp. Syst. **21**(1), 285–297 (2020)
18. Domino, K., Koniorczyk, M., Krawiec, K.,Jałowiecki, K., Gardas, B.: Quantum computing approach to railway dispatching and conflict management optimization on single-track railway lines. arXiv:2010.08227 (2021)
19. Costa, A.M.: A survey on benders decomposition applied to fixed-charge network design problems. Comput. Oper. Res. **32**(6), 1429–1450 (2005)
20. Paraskevopoulos, D.C., Bektaş, T., Crainic, T.G., Potts, C.N.: A cycle-based evolutionary algorithm for the fixed-charge capacitated multi-commodity network design problem. Eur. J. Oper. Res. **253**(2), 265–279 (2016)
21. Yaghini, M., Momeni, M., Sarmadi, M.: A simplex-based simulated annealing algorithm for node-arc capacitated multicommodity network design. Appl. Soft Comput. **12**(9), 2997–3003 (2012)
22. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
23. Rønnow, T.F., Wang, Z., Job, J., Boixo, S., Isakov, S.V., Wecker, D., Martinis, J.M., Lidar, D.A., Troyer, M.: Defining and detecting quantum speedup. Science **345**(6195), 420–424 (2014)
24. King, J., et al.: Quantum annealing amid local ruggedness and global frustration. J. Phys. Soc. Jpn. **88**(6), 061007 (2019)
25. D-Wave Systems has produced an open-source library in Python (dimod) for solvers that optimize QUBOs and Ising Hamiltonians. More information can be found here. https://docs.ocean.dwavesys.com/en/stable/docs_dimod/
26. Glover, F.: Tabu search–part I. ORSA J. Comput. **1**(3), 190–206 (1989)
27. Yarkoni, S., et al.: Quantum shuttle: traffic navigation with quantum computing, pp. 22–30. Association for Computing Machinery, New York (2020)