A Hybrid Solution Method for the Multi-Service Location Set Covering Problem

Irina Chiscop[1](✉), Jelle Nauta[1], Bert Veerman[1,2], and Frank Phillipson[1]

[1] The Netherlands Organisation for Applied Scientific Research, The Hague, The Netherlands
irina.chiscop@tno.nl

[2] Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

Abstract. The Multi-Service Location Set Covering Problem is an extension of the well-known Set Covering Problem. It arises in practical applications where a set of physical locations need to be equipped with services to satisfy demand within a certain area, while minimizing costs. In this paper we formulate the problem as a Quadratic Unconstrained Binary Optimization (QUBO) problem, apply the hybrid framework of the D-Wave quantum annealer to solve it, and investigate the feasibility of this approach. To improve the often suboptimal initial solutions found on the D-Wave system, we develop a hybrid quantum/classical optimization algorithm that starts from the seed solution and iteratively creates small subproblems that are more efficiently solved on the D-Wave but often still converge to feasible and improved solutions of the original problem. Finally we suggest some opportunities for increasing the accuracy and performance of our algorithm.

Keywords: D-Wave · MSLSCP · Optimization · Quantum annealing

1 Introduction

The past decade has seen the rapid development of the two paradigms of quantum computing, quantum annealing and gate-based quantum computing. In 2011 D-wave Systems announced the release of the world's first commercial quantum annealer[1] operating on a 128-qubit architecture, which has since been continually extended up to the 2048-qubit version, available from 2017[2]. These technological advances have led to a renewed interest in finding classical intractable problems suited for quantum computing.

[1] https://www.dwavesys.com/news/d-wave-systems-sells-its-first-quantum-computing-system-lockheed-martin-corporation.

[2] https://www.dwavesys.com/press-releases/d-wave%C2%A0announces%C2%A0d-wave-2000q-quantum-computer-and-first-system-order.

D-Wave's quantum processor is specifically designed to solve quadratic unconstrained binary optimization (QUBO) problems, and is therefore particularly suited for addressing NP-hard combinatorial optimization problems. Well-known examples that have been implemented on one of D-Wave's quantum processors include maximum clique [4], capacitated vehicle routing [6], minimum vertex cover [10], set cover with pairs [3], traffic flow optimization [9] and integer factorization [8]. These studies have shown that although the current generation of D-Wave annealers may not yet have sufficient scale, precision and connectivity to allow faster or higher quality solutions, they have the suitable infrastructure for modelling real-world instances of these problems, effectively decomposing these into smaller sub-problems and solving these on a real Quantum Processing Unit (QPU).

The paper at hand addresses another NP-hard combinatorial problem, the Multi-Service Location Set Covering Problem (MSLSCP) [11], arising in smart city planning. In smart cities, different services such as Wi-Fi, alarm and air quality or pollution sensors are integrated into street furniture like lamp posts and bus shelters, to create a dense network that can potentially achieve higher transmission rates and thus improve the quality of life. There are costs associated with both enabling a location to be equipped with services, and the actual equipping itself. The goal is then to distribute the services across the existing location in such a way that the demand for each service is satisfied at minimum total cost. As an extension of the Set Covering Problem, which is NP-hard [7], the MSLSCP is also NP-hard. Since for large instances this problem is computationally intractable, several heuristic solution methods have been proposed [11]. The best method found is based on distributing the services one-by-one over the available locations, and is therefore highly dependent on the order in which these services are considered. Given that large instances cannot be optimally solved classically, it is worthwhile to investigate how quantum annealing may be able to provide a better alternative to the current methods.

We present a novel algorithm to solve the MSLSCP, based on a two-phase hybrid approach. In the first phase, an initial solution is obtained from combining classical search heuristics and quantum annealing, whilst in the second phase, an improvement step is applied to reduce the size of the problem. This process is executed iteratively until the user-specified stopping criteria are met. As far as the authors know, this is the first time that the MSLSCP is modeled as a QUBO and solved by an algorithm employing quantum annealing.

The structure of this paper is as follows: Sect. 2 includes the mathematical integer linear programming and QUBO formulations of the problem. The hybrid functionality of the D-Wave system and the newly proposed approach are presented in Sect. 3. Section 4 explains how we generated the test problems to which we applied our method. The results obtained are discussed in Sect. 5. Finally, Sect. 6 provides some conclusions and ideas for further research.

2 Problem Formulation

In mathematical terms, the MSLSCP concerns the distribution of services $\mathcal{F} = \{1, \ldots, F\}$ over a set of locations $\mathcal{L} = \{1, \ldots, L\}$ such that the demand points $\mathcal{G}^u$ for each service $u \in \mathcal{F}$ are covered at minimal cost. Enabling a location $j \in \mathcal{L}$, incurs cost $f_j > 0$, whilst equipping a location $j \in \mathcal{L}$ with service $u \in \mathcal{F}$ incurs cost $c_j^u > 0$. An additional parameter a_{ij}^u is introduced to mark when a demand point falls within the range of a particular service placed at a particular location:

$$a_{ij}^u = \begin{cases} 1 & \text{if demand point } i \in \mathcal{G}^u \text{ is in range of location } j \in \mathcal{L} \text{ for service } u \in \mathcal{F}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The following binary decision variables are defined:

$$y_j = \begin{cases} 1 & \text{if location } j \in \mathcal{L} \text{ is open,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

$$x_j^u = \begin{cases} 1 & \text{if location } j \in \mathcal{L} \text{ is equipped with service } u \in \mathcal{F}, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The complete integer linear programming formulation of the MSLSCP is then given by:

$$\min \sum_{j \in \mathcal{L}} f_j y_j + \sum_{j \in \mathcal{L}} \sum_{u \in \mathcal{F}} c_j^u x_j^u, \tag{4}$$

$$\text{s.t.} \sum_{j \in \mathcal{L}} a_{ij}^u x_j^u \geq 1 \quad \forall i \in \mathcal{G}^u, \forall u \in \mathcal{F}, \tag{5}$$

$$x_j^u \leq y_j \quad \forall j \in \mathcal{L}, \forall u \in \mathcal{F}, \tag{6}$$

$$x_j^u \in \{0, 1\} \quad \forall j \in \mathcal{L}, \forall u \in \mathcal{F}, \tag{7}$$

$$y_j \in \{0, 1\} \quad \forall j \in \mathcal{L}. \tag{8}$$

The objective given by Eq. (4) represents the total sum of costs associated with enabling locations and equipping them with services. Equation (5) is a constraint enforcing all demand points to be satisfied. Constraint Eq. (6) expresses that a location must be enabled if it is equipped with any services. Finally, Eq. (7) and Eq. (8) specify that the opening and equipping variables should (of course) be binary.

To solve this problem on the D-Wave, we need to express the constraints (formulated above as inequalities) as equalities. Based on the formulation in Eq. (4)–(8) the MSLSCP can also be re-written as the following QUBO:

$$\min \quad A \cdot H_A + B \cdot H_B + C \cdot H_C, \tag{9}$$

where

$$H_A = \sum_{j\in\mathcal{L}} f_j y_j + \sum_{j\in\mathcal{L}}\sum_{u\in\mathcal{F}} c_j^u x_j^u, \tag{10}$$

$$H_B = \sum_{u\in\mathcal{F}}\sum_{i\in\mathcal{G}^u} \left(\sum_{j\in\mathcal{L}} a_{ij}^u x_j^u - \sum_{k=0}^{k_{i,u}^{\max}} 2^k \xi_{i,k}^u - 1\right)^2, \tag{11}$$

$$H_C = \sum_{j\in\mathcal{L}}\sum_{u\in\mathcal{F}} (x_j^u - x_j^u y_j). \tag{12}$$

H_A is the cost we aim to minimize described by Eq. (4).

H_B corresponds to the inequality constraint Eq. (5). Following the method described in [1], it uses slack variables $\xi_{i,k}^u$, effectively adding equations to capture each possible way of satisfying the inequality. The simplest way in which Eq. (5) can be satisfied is by having only one non-zero $a_{ij}^u x_j^u$ combination, in which case the sum with the $\xi_{i,k}^u$'s should be zero and the entire Eq. (5) is minimized to zero. If there are more non-zero $a_{ij}^u x_j^u$ combinations (i.e. the demand point is serviced multiple times), we can change the appropriate $\xi_{i,k}^u$'s to again minimize the total to zero. The number of additional variables introduced, for each of the $\sum_{u\in\mathcal{F}} |\mathcal{G}^u|$ constraints of type H_B, is given by:

$$k_{i,u}^{\max} = \lfloor \log_2 (\sum_{j\in\mathcal{L}} a_{ij}^u - 1) \rfloor. \tag{13}$$

The reason we only need logarithmically many slack variables is that they act as bits in the binary representation of the number of ways in which we can satisfy the inequality Eq. (5).

H_C corresponds to constraint Eq. (6). This term is minimized if services are placed only at open locations.

The penalty coefficients A, B, C should be set such that the minimum of the objective in (9) satisfies the constraints (i.e., we cannot achieve a lower minimum by breaking constraints in favor of lowering the cost in Eq. (10)). In practice there is a delicate trade-off here. Setting lower B and C causes constraints to be violated for many sub-optimal solutions found, while setting higher B and C ensures that the constraints are satisfied but leads to lower accuracy in optimizing the actual cost of Eq. (10). Through some experimental tests we concluded that a suitable parameter setting is:

$$A = 1, \tag{14}$$

$$B = 2 \cdot (\max\{f_j : j \in \mathcal{F}\} + \max\{c_j^u j \in \mathcal{L}, u \in \mathcal{F}\}), \tag{15}$$

$$C = 2 \cdot (\max\{c_j^u : j \in \mathcal{L}, u \in \mathcal{F}\}). \tag{16}$$

3 Solution Approach

To solve the MSLSCP we propose a hybrid iterative approach, which combines a method to reduce the size of the problem together with D-Wave's built-in hybrid framework. First, the process of quantum annealing and D-Wave's hybrid capabilities are presented. Then, a detailed overview of the newly proposed algorithm is given.

3.1 Quantum Annealing on D-Wave

The devices produced by D-Wave Systems are practical implementations of quantum computation by adiabatic evolution [5]. The evolution of a quantum state on D-Wave's QPU is described by a time-dependent Hamiltonian, composed of initial Hamiltonian H_0, whose ground state is easy to create, and final Hamiltonian H_1, whose ground state encodes the solution of the problem at hand:

$$H(t) = \Big(1 - \frac{t}{T}\Big)H_0 + \frac{t}{T}H_1. \tag{17}$$

The system in Eq. (17) is initialized in the ground state of the initial Hamiltonian, i.e. $H(0) = H_0$. The adiabatic theorem states that if the system evolves according to the Schrödinger equation, and the minimum spectral gap of $H(t)$ is not zero, then for time T large enough, $H(T)$ will converge to the ground state of H_1, which encodes the solution of the problem. This process is known as quantum annealing. Although here we are not concerned with the technical details, it is worthwhile to mention that it is not possible to estimate an annealing time T to ensure that the system always evolves to the desired state. Since there is no estimation of the annealing time, there is also no optimality guarantee.

The D-Wave quantum annealer can accept a problem formulated as an Ising Hamiltonian, corresponding to the term H_1 in Eq. (17), or rewritten as its binary equivalent, in QUBO formulation. Next, this formulation needs to be embedded on the hardware. In the most developed D-Wave 2000Q version of the system, the 2048 qubits are placed in a Chimera architecture: a 16×16 matrix of unit cells consisting of 8 qubits. This allows every qubit to be connected to at most 5 or 6 other qubits. With this limited hardware structure and connectivity, fully embedding a problem on the QPU can sometimes be difficult or simply not possible. In such cases, the D-Wave system employs built-in routines to decompose the problem into smaller sub-problems that are sent to the QPU, and in the end reconstructs the complete solution vector from all sub-sample solutions. The first decomposition algorithm introduced by D-Wave was *qbsolv* [2], which gave a first possibility to solve larger scale problems on the QPU. Although *qbsolv* is the main decomposition approach on the D-Wave system, it does not enable customizations, and therefore is not particularly suited for all kinds of problems.

To alleviate the short-comings of *qbsolv*, D-Wave introduced a hybrid framework which enables users to quickly design and test workflows that iterate over sets of samples through different samplers to solve arbitrarily-sized QUBOs. Large problems can be decomposed in different ways and two or more solution techniques can run in parallel[3]. The schematic representation is shown in Fig. 1. There are four branches shown in this example, more precisely a classical

[3] https://readthedocs.com/projects/d-wave-systems-dwave-hybrid/.

Interruptable Tabu search and three decomposition-based methods. A workflow branch has the following structure: decomposer - sampler - composer. We briefly explain the purpose of each these building blocks.

A decomposer is a component that splits up the original problem into subproblems by selecting only a part of the variables. Classical decomposition approaches are: select first n variables that have the highest impact on your objective (energy-based selection) or select variables that show up in the same set of constraints together (constraint-based selection). Here, *qbsolv* is an example of an energy-based selection algorithm. The sampler is the chosen method to sample (solve) the subproblems coming from the decomposer. This can be simulated annealing or any other QPU-based sampler. Lastly, a composer makes a selection from current samples and updates the complete, final solution according to user-defined criteria.

This workflow allows custom design of the different building blocks and makes it possible to combine these into different methods that can be run in parallel as can be seen in Fig. 1. In our numerical experiments, we use *Kerberos*, the reference hybrid built-in sampler, which combines Tabu search, simulated annealing, and D-Wave sub-problem sampling on problem variables that have high-energy impact.

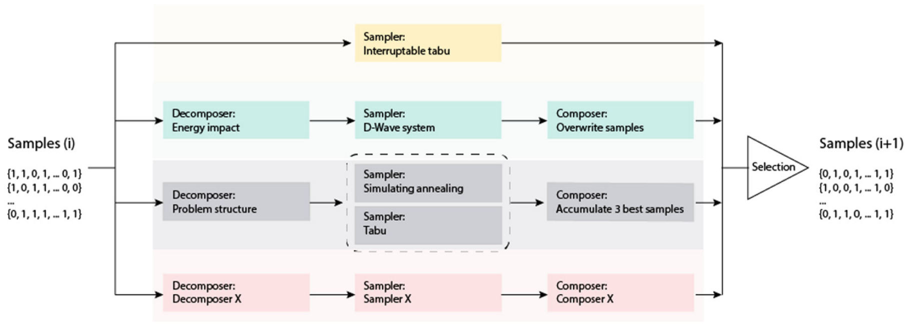

Fig. 1. Schematic representation of D-wave hybrid workflow (see footnote 3).

3.2 Two-Stage Hybrid Algorithm

Using the *Kerberos* sampler from the D-wave hybrid framework, we often cannot find even a feasible - let alone optimal - solution to our problem within a reasonable time. To counter this limitation we propose an hybrid approach in which we make use of the initially found solutions (which may be infeasible) to restrict the solution space and then re-solve the problem. This approach is described in Algorithm 1.

The idea of the algorithm is to construct a first solution by overlapping different samples obtained from Kerberos. In this way we increase the probability of creating a feasible solution. The desired number of solutions to be overlapped

```
1  $K_{max}$ = number of runs
2  $N_{max}$ = number of improvement iterations
3  instance = $\mathcal{L}, \mathcal{F}, \mathcal{G}^u \quad \forall u \in \mathcal{F}$
4  solutions = $\{\}$
5  for $k = 1, \ldots, K_{max}$ do
6      solution$_k$ ← KerberosSampler(instance);
7      solutions ← solutions ∪ solution$_k$
8  end
9  solution ← Overlap(solutions)
10 for $N = 1, \ldots, N_{max}$ do
11     if solution is feasible then
12         $\mathcal{L} \leftarrow \{j : y_j = 1\}$
13         instance ← $\mathcal{L}, \mathcal{F}, \mathcal{G}^u \quad \forall u \in \mathcal{F}$
14         solution ← KerberosSampler(instance);
15     else
16         for $u \in \mathcal{F}$ do
17             $\tilde{\mathcal{G}}_u = \{i \in \mathcal{G}_u : \sum_{j \in \mathcal{L}} a_{ij}^u x_j^u = 0\}$
18         end
19     end
20     $\mathcal{L} \leftarrow \{j : y_j = 1 \text{ or } a_{ij}^u = 1 \quad \forall u \in \mathcal{F}, \quad \forall i \in \tilde{\mathcal{G}}_u\}$
21     instance ← $\mathcal{L}, \mathcal{F}, \mathcal{G}^u \quad \forall u \in \mathcal{F}$
22     solution ← KerberosSampler(instance)
23 end
24 return solution
```

Algorithm 1: Hybrid algorithm pseudocode.

is given as input (step 1). The problem instance to be solved is specified by the sets of locations, services and demand points (step 3). Once the seed solution is created by combining the Kerberos-solutions (steps 5–9), the algorithm moves to the iterated improvement stage. In each iteration, there are two possibilities:

1. If the last solution found was feasible, the problem space is restricted by removing the set of unopened locations from the original input (steps 11–15). This step is motivated by the fact that in a feasible solution, some locations may be opened unnecessarily and the services may be equipped more efficiently on the opened locations.
2. If the last solution found was infeasible, the location set is again reduced (steps 16–18). The infeasible solutions observed were all caused by the violation of demand point coverage constraints. As such, the locations which remain unused in the solution and which cannot cover any of the unsatisfied demand points, can be removed from the location set (step 19).

The newly produced instance can then be solved with Kerberos again and the improvement step process can be repeated (steps 20–21). While these simple strategies of reducing the search space may not always result in optimal solutions, they provide a means to find a reasonably good feasible solution.

4 Problem Generation

To apply our algorithm on a test set, we generate problem instances with the numbers of services, locations and demand points as input parameters. The generation is done in two steps as explained below. For more details on some of the steps, see the code on Github[4].

First we generate the set of coordinates in the unit square at which demand points and locations are situated. This is done in such a way that they are not too close to each other, each demand point is reachable from at least one location, and each location can service at least one demand point.

We then assign each demand point a single requested service, in such a way that each service is requested at least once (see Algorithm 2).

```
$\mathcal{F}$ = the set of $F$ services
$\mathcal{P}_L$ = Location points
$\mathcal{P}_U$ = Demand points
triplets $\mathcal{T} = \{\}$
$f$ = random service in $\mathcal{F}$
for $u \in \mathcal{P}_U$ do
    $L_U = \{l \in \mathcal{P}_L : \text{distance}(u, l) \leq d_{max}\}$
    for $l \in L_U$ do
        $\mathcal{T} \leftarrow \mathcal{T} \cup (f, l, u)$
        if $\exists (f, l, u) \in \mathcal{T}, l \in \mathcal{P}_L, u \in \mathcal{P}_U \forall f \in \mathcal{F}$ then
            $f \leftarrow$ random service in $\mathcal{F}$
        else
            $f \leftarrow$ some service in $\mathcal{F} : (f, l, u) \notin \mathcal{T}\ \forall l \in \mathcal{P}_L, \forall u \in \mathcal{P}_U$
        end
    end
end
```

Algorithm 2: Pseudocode to assign service requirements to demand points by generating triplets $(f, l, u) \in \mathcal{F} \times P_L \times P_U$.

In this way we generate a set of problems of various sizes denoted FLU, where F is the number of services, L is the number of locations and U is the number of demand points. The generated problem and the size of their QUBO formulation, given by the amount of binary variables in the solution vector, is shown in Table 1. Given that on the current 2048 qubit architecture one can embed a fully-connected graph of about 60 nodes, it is clear that most of the problems listed in Table 1 cannot be directly mapped to the QPU and will have to be decomposed.

[4] https://github.com/jcnauta/DWave-MSLSCP.

Table 1. Overview of generated problems.

Problem parameters	QUBO size	Problem parameters	QUBO size
F2L50U50	321	F4L50U200	942
F4L50U50	416	F2L100U200	1092
F2L50U100	483	F4L100U200	1274
F2L100U50	498	F2L200U200	1462
F4L50U100	585	F2L50U400	1554
F4L100U50	685	F4L50U400	1626
F2L100U100	691	F2L100U400	1874
F2L50U200	825	F4L100U400	2044
F4L100U100	888	F2L200U400	2340

5 Results

In this section we present the results of our hybrid method described by Algorithm 1. We look first at the quality of the solution obtained in the first sampling phase, and then assess the performance of the iterative improvement step in the second phase.

5.1 First Phase Results

Table 2 shows the results obtained in the first phase of Algorithm 1, in which all problem instances are sampled using the D-Wave's built-in hybrid framework. The Kerberos sampler was run 50 times for each problem, with $time\_out = 60\,\text{s}$ and $max\_iter = 10$ as stopping criteria. The rest of the parameters were set to the default values. We assessed the quality of solutions in terms of the deviation from the optimum solution, which was classically obtained from each problem using the CPLEX commercial solver, version 12.8. A striking observation to emerge from these results is that with the exception of the smallest problem, F2L50U50, the average deviation of the 50 runs from optimum is quite large ranging from 20% to over 200%. Moreover, for most problems, the amount of feasible solutions out of the 50 runs was quite small, whilst in a few cases no feasible solution could be found at all. Interestingly, these correspond to problems in which the number of demand points greatly exceeds the number of locations to be equipped with services. A possible explanation is that there are relatively fewer ways to feasibly allocate the services when the number of locations is lower.

Table 2. Phase 1 results for each of the test instances. The entries marked with "-" in the second column indicate that no feasible solution was found for that particular problem.

Problem	Deviation (%) of best feasible solution found of 50 runs	Avg. deviation (%) of 50 runs from optimum	Number of feasible solutions found of 50 runs
F2L50U50	1.87	16.3	41
F4L50U50	54.36	73.8	48
F2L50U100	20.23	30.01	3
F2L100U50	33.36	46.28	50
F4L50U100	123.94	83.98	1
F4L100U50	67.67	97.02	43
F2L100U100	41.13	53.34	8
F2L50U200	-	51.1	0
F4L100U100	87.66	99.84	17
F4L50U200	-	116.21	0
F2L200U100	106.55	66.34	1
F2L100U200	-	30.15	0
F4L200U50	111.61	97.78	1
F4L100U200	-	77.31	0
F4L50U400	-	116.67	0
F2L100U400	-	107.48	0
F4L100U400	-	209.45	0
F2L200U400	256.59	228.83	1

Figure 2 depicts the deviation from the optimal solution of each problem. We see that as the problem size increases, the variance of the results also increases. Once again, there seem to be higher fluctuations in the solutions of problems with fewer locations than demand points. The results obtained so far indicate that the quality of the solutions obtained with Kerberos is rather low. This might be improved by modifying the utilized stopping criteria, but our iterated improvement method (phase 2) addresses this problem automatically and quite reliably.

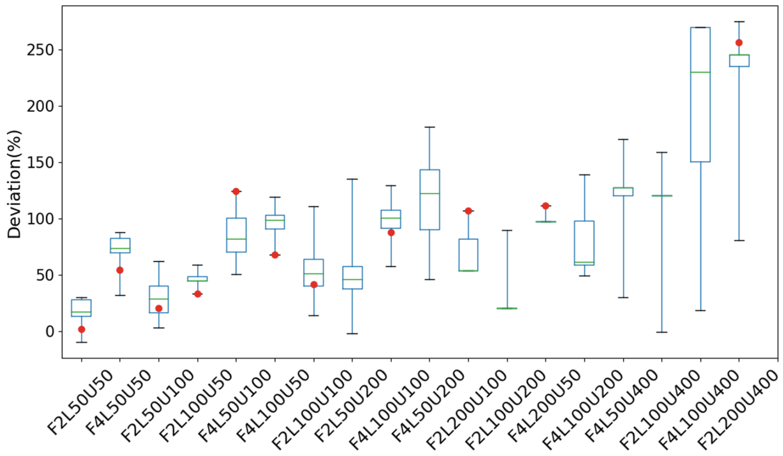

Fig. 2. Deviation of the results from the optimal solution for each problem. Each box corresponds to the area in which the middle 50% of the data reside in with the continuous line being the median. The whiskers are extended to the minimum and maximum values of the data. The red dot marks the best feasible solution found for the respective problem. (Color figure online)

5.2 Improvement Phase Results

In the second phase of Algorithm 1, starting at step 9, the 50 solutions obtained in the first phase were overlapped, resulting in an initial solution to be used as input for the iterated improvement. This second component of the algorithm was run for 10 iteration using both Kerberos and Simulated Annealing (SA) with the same parameter setting for sampling (in step 14 of the algorithm) to ensure a fair comparison between the two. The results of the second phase are summarized in Table 3. The second and third columns of this table show the minimum deviation from the optimum over all iterations resulting in a feasible solution. The results clearly show that with the exception of problem F2L50U50, the improvement step is effective, resulting in either better solutions or finding feasible solutions where Phase 1 did not. Another interesting observation is that Kerberos outperforms SA in Phase 2, almost always achieving solutions with a smaller optimality gap. This is expected, since Kerberos benefits from the additional sampling on the quantum annealer. Exceptional cases are problems F4L50U400, F2L100U400 and F4L100U100 to which Kerberos, unlike SA, fails to find a feasible solution within 10 iterations.

If we now turn to the performance of the second phase in terms of solution improvement, shown in Fig. 3, we observe large variations across different problems. For small-sized problems (Fig. 3a), the initial solution of the second phase is most often feasible, and the optimality gap reduces significantly in the first four or five iterations when using Kerberos. However, it occurs that if the deviation from the optimum becomes small, than the corresponding assignment of services to locations is less flexible, leading to no further improvement in

Table 3. Phase 2 results for each of the test instances. The entries marked with "-" in the second column indicate that no feasible solution was found for that particular problem.

Problem	Deviation (%) of best feasible solution found in Phase 1	Deviation (%) of best feasible solution found in Phase 2 (Kerberos)	Deviation (%) of best feasible solution found in Phase 2 (SA)
F2L50U50	1.87	2.45	18.99
F4L50U50	54.36	40.47	43.24
F2L50U100	20.23	16.07	17.04
F2L100U50	33.36	23.01	23.01
F4L50U100	123.94	21.43	57.60
F4L100U50	67.67	28.80	31.56
F2L100U100	41.13	24.96	26.58
F2L50U200	-	13.95	25.37
F4L100U100	87.66	15.50	42.43
F4L50U200	-	196.34	207.05
F2L200U100	106.55	19.06	36.34
F2L100U200	-	62.04	86.33
F4L200U50	111.61	63.25	57.73
F4L100U200	-	29.33	97.52
F4L50U400	-	-	242.80
F2L100U400	-	-	117.79
F4L100U400	-	-	252.97
F2L200U400	256.60	102.96	156.08

the latter iterations. This behaviour is also observed for some problems when employing SA (Fig. 3b), although in this case, the solution at each iteration may drastically vary, and occasionally become infeasible. For the largest problems, none of the two samplers seem to be very effective since most iterations result in infeasible solutions (Fig. 3c and Fig. 3d). This may be due to the fact that the overlap solution obtained from the 50 solutions in the first phase is infeasible, and does not incorporate enough open locations to allow for an improved search throughout the iterations. To this end, a better initial solution could be obtained in the first phase if we would allow for more than 50 runs.

The results obtained so far suggest that small problems can be solved relatively well with our hybrid approach, when using Kerberos in both phases. The solutions for the larger problems, for which the number of demand points greatly exceeds the number of available locations, are of lower quality. These problems could benefit in particular by a different approach to determine the starting solution. Instead of overlapping all solutions from the first phase, one

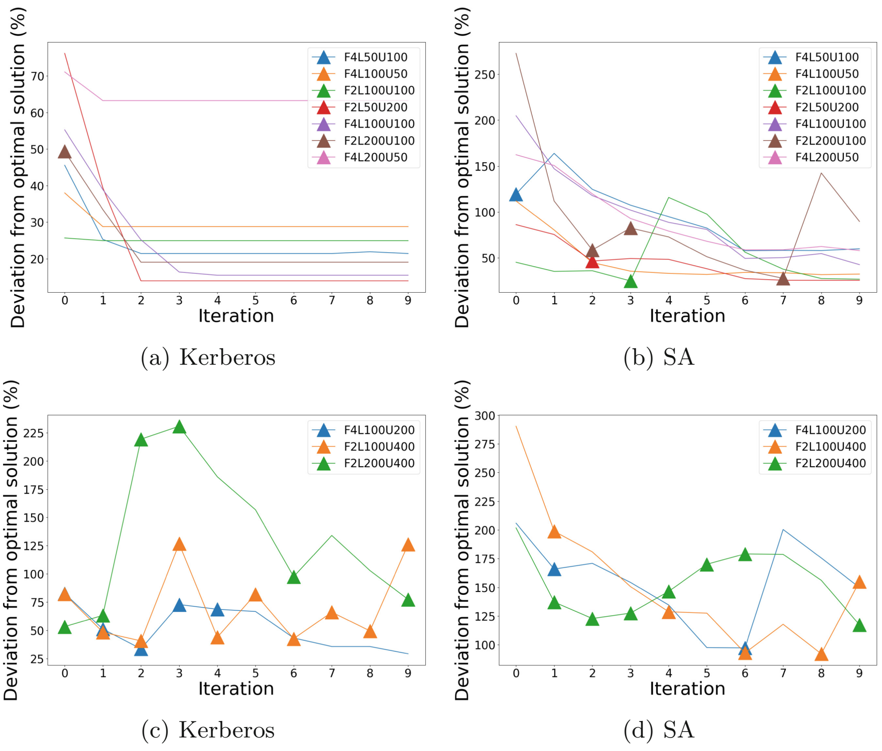

Fig. 3. Phase 2 results per iteration for different problems resulting in feasible solutions (a and b) and infeasible solutions (c and d). The sub-figure captions indicate the sampler used. The triangle markers indicate infeasible solutions.

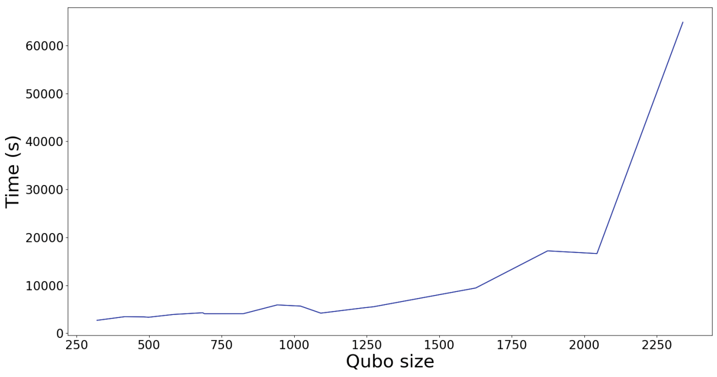

Fig. 4. Wall-clock time taken by Algorithm 1 as a function of the problems' qubo size.

could look into combining solutions which are sufficiently "different", i.e., which have ideally dissimilar sets of open locations. Furthermore, all problems could benefit from further parameter tuning in the Kerberos sampler and increasing the number of iterations in the improvement step. The current stopping criteria for Kerberos and limited number of limitations were chosen so as to provide a first proof of concept of our hybrid approach. Even in the current setting, the wall-clock time recorded for each problem was in the range of 2–5 h with some of the larger problems far exceeding this interval, as shown in Fig. 4. However, the QPU time needed for each of the decomposed subproblems ranged in the interval 10–300 ms, whilst the cumulated recorded QPU time for all experiments was less than 5 min. This is still underperforming in comparison to the classical solver, CPLEX, for which the average problem run time was 0.16 s. Since our experiments were run using a cloud service, it is likely that the high computational times shown in Fig. 4 are due to the latency between submitting problems to the QPU and receiving the solution. In the current D-Wave environment it is difficult to collect timing information for individual problems and therefore, we cannot estimate the queuing or read-out time.

6 Conclusion

This paper introduced a very first attempt to design a hybrid approach to solve the MSLSCP, by modelling the problem as a qubo and employing D-Wave's hybrid framework in combination with a classical improvement step. To the best of our knowledge, this work is the first to address this problem by combining classical means and quantum annealing. It was shown that the two-phase hybrid algorithm leads to improved feasible solutions compared to Kerberos sampling for almost all problems. Although the proposed approach does not yield a direct improvement in terms of solutions quality or running time with respect to classical solvers, it provides a way to model the problem in a suitable way and to reduce the size of the search space iteratively, in the attempt to overcome the physical limitations of the current quantum annealers. As some of the problem instances tackled in this paper far exceeded the size of the available QPU, we would expect that with the development and extension of the current hardware architecture, the performance of the hybrid tools will increase.

The findings of this study suggest several courses of action for improving the performance and results of Algorithm 1. A first intuitive step is to repeat the experiment shown in this paper with adjusted parameters for Kerberos. It is likely that modifying the internal Kerberos loop parameters or simply adjusting the stopping criteria will result in better solutions. Moreover, a reasonable approach to tackle the infeasibility of large problems is to propose a different way to generate a starting solution for the improvement step of the algorithm, which needs to be sufficiently diverse. This could be achieved by implementing more Kerberos runs in the first phase or applying some classical greedy heuristics.

Finally, we expect that intrinsic problem structure given by the spatial distribution of demand points heavily influences the performance of the algorithm. Therefore, it is worthwhile to investigate ways in which the decomposition of the problem on the QPU could be customized to incorporate this aspect.

References

1. D-wave problem-solving handbook; reformulating a problem. https://docs.dwavesys.com/docs/latest/c_handbook_3.html. Accessed 27 Jan 2020
2. Booth, M., Reinhardt, S.P., Roy, A.: Partitioning optimization problems for hybrid classical/quantum execution. Technical report, D-Wave Systems, September 2017
3. Cao, Y., Jiang, S., Perouli, D., Kais, S.: Solving set cover with pairs problem using quantum annealing. Sci. Rep. **6**(1), 1–15 (2016)
4. Chapuis, G., Djidjev, H., Hahn, G., Rizk, G.: Finding maximum cliques on the D-Wave quantum annealer. J. Sign. Process. Syst. **91**(3), 363–377 (2018). https://doi.org/10.1007/s11265-018-1357-8
5. Farhi, E., Goldstone, J., Gutmann, S., Sipser, M.: Quantum computation by adiabatic evolution. arXiv:quant-ph/0001106v1 (2000)
6. Feld, S., et al.: A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer. Front. ICT **6**, 13 (2019)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
8. Jiang, S., Britt, K.A., McCaskey, A.J., Humble, T.S., Kais, S.: Quantum annealing for prime factorization. Sci. Rep. **8**(1), 1–9 (2018)
9. Neukart, F., Compostella, G., Seidel, C., Dollen, D.V., Yarkoni, S., Parney, B.: Traffic flow optimization using a quantum annealer. Front. ICT **4**, 29 (2017)
10. Pelofske, E., Hahn, G., Djidjev, H.: Solving large minimum vertex cover problems on a quantum annealer. In: Proceedings of the 16th ACM International Conference on Computing Frontiers, CF 2019, pp. 76–84. ACM, New York (2019)
11. Phillipson, F., Vos, T.: Dense multi-service planning in smart cities. In: International Conference on Information Society and Smart Cities (2018)