Article

A QUBO model for the Traveling Salesman Problem with Time Windows for execution on the D-Wave

Christos Papalitsas*, Theodore Andronikos*, Konstantinos Giannakis*, Georgia Theocharopoulou, and Sofia Fanarioti

Department of Informatics, Ionian University, Tsirigoti Square 7, Corfu, 49100, Greece; {kgiann, c14papa, zeta.theo, sofiafanar, andronikos}@ionio.gr

* Correspondence: c14papa@ionio.gr (Ch.P.); andronikos@ionio.gr (Th.A.); kgiann@ionio.gr (K.G.);

‡ These authors contributed equally to this work.

Abstract: This work focuses on expressing the TSP with Time Windows (TSPTW for short) as a quadratic unconstrained binary optimization (QUBO) problem. The time windows impose time constraints that a feasible solution must satisfy. These take the form of inequality constraints, which are known to be particularly difficult to articulate within the QUBO framework. This is, we believe, the first time this major obstacle is overcome and the TSPTW is cast in the QUBO formulation. We have every reason to anticipate that this development will lead to the actual execution of small scale TSPTW instances on the D-Wave platform.

Keywords: TSP; TSPTW; Metaheuristics; Quantum Annealing; Ising Model; QUBO; D-Wave

1. Introduction

Quantum computing promotes the exploit of quantum-mechanical principles for computation purposes. Richard Feynman in 1982 suggested that computation could be done more efficiently by taking advantage of the power of quantum "parallelism" [1]. Since then quantum computing has gained a lot of momentum and today appears to be one of the most prominent candidates to partially replace classical, silicon-based systems. There exist a number of quantum algorithms that run more efficiently than the best known classical algorithm; arguably, the most famous ones are Shor's [2] and Grover's [3] algorithms. For an in-depth study of quantum computing and quantum information, the reader is referred to [4].

When solving large scale optimization problems, one of the most effective classical strategies is to search among the nearest neighbors and search for paths between the initial and final configurations to improve upon an initial guess y_t. This search takes place locally among neighboring configurations similar to y_t. By finding the best solution within the local neighborhood of y_t, the next candidate solution y_{t+1} appears. Then a new local search starts with y_{t+1} as the starting point in the neighborhood. Unfortunately, greedy local improvement, in case of hard problems, may be deceptively leading the solution into a local minima whose energy may be much higher than the globally minimum value.

Adiabatic quantum computation was proposed by Farhi et al. [5,6] in the early 2000s [7] and is based on an important theorem of quantum mechanics, the adiabatic theorem [8,9]. This theorem dictates that if a quantum system is driven by a slowly changing (in time) Hamiltonian that evolves from H_{init} to H_{fin}, then if the systems starts in the ground state of H_{init}, the system will end up in the ground state of H_{fin}. Farhi et al. also showed that adiabatic quantum computation can be efficiently simulated by a quantum computer based on the gate model, meaning that the two computational models have the same expressive power [5,6].

Quantum annealing is based on the quantum adiabatic computation paradigm. Quantum annealing was initially proposed by Kadowaki and Nishimori [10,11] and has since then been used for tackling combinatorial optimization problems. The way that a quantum annealer tries to solve problems, is very similar to the way optimization problem are solved using (classical) simulated annealing [12]. An energy landscape is constructed, via a multivariate function, such that the ground state's coordinates (i.e., the lowest value) correspond to the solution of the problem. The quantum annealing process is iterated to the point that an optimal solution, with a sufficient high probability, is found. The "quantum" in "quantum annealer" refers to the use of multi qubit tunneling [10,11]. The high degree of parallelism is the advantage of quantum annealing over classical code execution. A quantum annealer explores all possible inputs in parallel to find the optimal solution to a problem, which might prove crucial when it comes to solving NP-complete problems. We caution the reader however that, at present, these techniques should be considered more as an automatic heuristic-finding program than as a formal solver [13].

The preceding facts explain why optimization problems have been associated with quantum computing principles [14–18]. In this approach, which can be considered as an "analog" method to tackle optimization problems, the ground state of a Hamiltonian represents an optimal solution of the optimization problem at hand. Typically, the quantum annealing process starts with the system being in an equal superposition of all states. Then, an appropriate Hamiltonian is applied and the system evolves in a time-dependent manner according to the Schrödinger equation. The state of the system keeps changing according to strength of the local transverse field, which varies with respect to time. Finally, the transverse field is smoothly turned off and the system finds itself in the ground state of a properly chosen Hamiltonian that encodes an optimal solution of the optimization problem.

After the first commercial launch of an actual system whose functionality is based upon quantum annealing, the D-Wave platform, it has been demonstrated that these particular quantum machines are capable of solving certain quadratic unconstrained (mainly binary) optimization problems. The core of D-Wave's machine that applies the quantum annealing principles for complex combinatorial optimization problems is the quantum processing unit (QPU). In the D-Wave computer the quantum bits (which we shall refer to as qubits from now on) are the lowest energy states of superconducting loops [7,19,20]. In these states there is a circulating current and a corresponding magnetic field. Since a qubit is a quantum object, its state can be a superposition of the 0 state and the 1 state at the same time. However, upon measurement a qubit collapses to the state 0 or 1 and behaves like a classical bit. The quantum annealing process in effect guides the qubits from a superposition of states to their collapse into either the 0 or 1 state. In the end, the net effect is that the system is in a classical state, which must encode an (optimal) solution of the problem.

The current generation of D-Wave computers employs the Chimera topology. In the Chimera topology, qubits are sets of connected unit cells, that are connected to four vertical qubits via couplers [20–22]. The unit cells are oriented vertically and horizontally with adjacent qubits connected, creating a network of sparsely connected qubits. A Chimera graph consists of an $N \times N$ grid of unit cells. The D-Wave 2000Q QPU has up to 2048 qubits which are mapped to a C16 Chimera graph, that is they are logically mapped into a 16×16 matrix of unit cells, each consisting of 8 qubits. In the D-Wave nomenclature the percentage of working qubits and couplers is known as the working graph, which is typically a subgraph of the total number of interconnected qubits, which are physically present in the QPU.

A major category of optimization problems, particularly amenable to D-Wave's quantum annealing, are those that can be expressed as quadratic unconstrained binary optimization (QUBO) problems. QUBO refers to a pattern matching technique, that, among other applications, can be used in machine learning and optimization, and which involves minimizing a quadratic polynomial over binary variables [20,23–29]. We emphasize that QUBO is NP-hard [29]. Some of the most famous combinatorial optimization problems that can be solved as QUBO problems are the Maximum Cut, the Graph Coloring and the Partition problem [30]. More details on the QUBO formulation and related results (that are beyond the scope of this paper) can be found in the survey paper of Kochenberger [25]. QUBO is equivalent to the Ising model, a well-known and extensively studied model in physics, that was introduced in the mid 1920s by Ernst Ising and Wilhelm Lenz in the field of ferromagnetism [31,32]. The underlying logical architecture of this model is that variables are represented as qubits and interactions among qubits stand for the costs associated with each pair of qubits. In particular, this architecture can be depicted as an undirected graph with qubits as vertices and couplers as edges among them. The open-source software qbsolv that D-Wave introduced in 2017 is aimed at tackling QUBO problems of higher scale than previous attempts, by utilizing a more complex graph structure with higher connectivity among QPUs, by partitioning the input into parts that are then independently solved. This process is repeated using a Tabu-based search until no further improvement is found [19,33].

The literature contains several works that are dedicated to solving the standard TSP or some relative problem in a quantum setting. One of the first, was the work by Martoňák et al. in [34] that introduced a different quantum annealing scheme based on path-integral Monte Carlo processes to address the symmetric version of the Traveling Salesman Problem (sTSP). In [35,36] the D-Wave platform was used as a test bed for evaluating the efficiency of quantum annealing in solving the standard TSP compared to classical methods. In [37,38] the well-known variation of the TSP, the (Capacitated) Vehicle Routing Problem is studied using, again, the D-Wave computer. However, no work is known to us that tackles the TSP with Time Windows within the QUBO framework, using the D-Wave platform, or even quantum annealing in general.

Contribution. In this paper we give the first, to the best of our knowledge, QUBO formulation for the TSP with Time Windows (TSPTW). The existence of an Ising or QUBO formulation for a problem is the essential precondition for its solution on the current generation of D-Wave computers. For the vanilla TSP there exists such a formulation, as presented in an elegant and comprehensive manner in [32], which has enabled the actual solution of TSP instances on the D-Wave platform (see [35–37]). In contrast, prior to this work, the TSPTW had not been cast in the QUBO framework. This can be attributed to the extra difficulty of expressing the time window constraints of TSPTW. We hope and expect that the formulation presented here will lead to the experimental execution of small-scale TSPTW instances.

This paper is organized as follows. The most relevant to our study work is presented in Section 2. Section 3 is devoted to the standard definitions and notation of the conventional Traveling Salesman Problem with Time Windows. Section 4, the most extensive section of this article, contains the main contribution of our paper. It includes an in-depth presentation of all the required rigorous mathematical definitions and the proposed modeling that allows us to map the TSPTW into the QUBO framework. Finally, conclusions and ideas for future work are given in Section 5.

2. Related work

Quantum annealing [5,11] has been shown to provide solutions to a broad range of combinatorial optimization problems, not only in computer science [6,16,26,30,39–41], but also in other fields, such as quantum chemistry [42], protein folding [14], vehicle routing [33,38,43,44], etc. This kind of problems aim at minimizing a cost function, which can be interpreted as finding the ground state of a typical Ising Hamiltonian [32]. Nevertheless, it is a laborious task to compute a global minimum in problems where multiple local minima exist [45–47], a fact that shares a lot of similarities to classical spin glasses [45,46].

The possible superiority of quantum computation could be translated into either providing a better solution (i.e., closest to the optimal one) or arriving at a solution faster or producing a diverse set of solutions (for the multiobjective case). Some known cases where such methods work well are spin glasses [48], graph coloring [49], job-shop scheduling [50], machine learning [16,51], graph partitioning [30], 3-SAT [52], vehicle routing and scheduling [33,38,43,44], neural networks [53], image processing citecruz2018qubo etc., where the problem parameters can be expressed as boolean variables.

Battaglia et al. showed that quantum annealing techniques could outperform their classical counterparts on a known NP-complete problem, the 3-SAT, under special circumstances, whereas in the general case, the quantum versions did not offer any actual advantage [52]. In a recent work, Pagano et al. built a mechanism that implements a shallow-depth quantum approximate optimization algorithm (QAOA) by estimating the ground state energy of the Ising model using an analogue quantum simulator [54]. Farhi and Harrow tried to show the advantages of quantum approximate optimization algorithms compared to classical approaches, providing useful theoretical results and bounds, with the emphasis on the idea of "quantum supremacy" than can be established through such solutions [55]. Constrained polynomial optimization problems using adiabatic quantum computation methods were recently discussed by Rebentrost in [18]. In [56], Venegas-Andraca et al. introduced basic algorithms for some well-known problems in combinatorial optimization, like the 3-SAT [52] and the max-cut [57] problems. An overview of approaches of the quantum annealing systems used by D-Wave Systems [19] is, also, presented.

Recently, in [58], Hadfield et al. worked upon the quantum annealing algorithm of Farhi et al. [6], extending it by employing the quantum alternating operator ansatz, which yields a broader set of operators that can be used by the user. Particularly, this operator allows the representation of a larger set of states compared to the original algorithm, aiming to tackle problems with tighter constraints. Another work on how to apply constraints in QUBO schemes was presented by Vyskocil and Djidjev in [59]. In particular, to avoid the use of large coefficients (hence, more qubits) that result from the use of quadratic penalties, they proposed a novel combinatorial design and solving of mixed-integer linear programming problems to accommodate the application of the desired constraints.

Choi in [7], one of the first studies regarding the commercial D-Wave machine, showed that quadratic unconstrained binary optimization (QUBO) problems can be solved using an adiabatic quantum computer that employs an Ising spin-1/2 Hamiltonian. This was achieved by the reduction, through minor-embedding, of the underlying graph to the quantum hardware graph. The Chimera graph is the underlying annealing architecture of the current generation D-Wave platform. Due to physical limitations and noise levels, some qubits and couplers cannot be exploited, and are, thus, disabled. Therefore, the underlying graph is marginally incomplete [21,22].

In a recent technical report, D-Wave systems describe in detail their next generation architecture graph, named Pegasus [20]. As claimed by D-Wave itself, Pegasus will offer more flexibility and expressiveness over previous topologies, like more efficient embeddings of cliques, penalties, improved run times, boosted energy scales, better handling of errors, etc [20]. Similarly, Dattani and Chancellor discussed some differences between the two latest quantum annealing architectures from D-Wave systems, namely the Chimera and Pegasus graphs [22]. They further proposed a methodology to minor embed the required subgraphs on the Chimera and Pegasus graphs.

The D-Wave Two, 2X, and 2000Q all used the Chimera graph (see Table 1), which consisted of processing unit of $K_{4,4}$ subgraphs. Each generation of this graph has evolved by exploiting more and more qubits (or vertices). On the other hand, Pegasus, the latest graph, totally changed the setting by adding more complex connectivity (each qubit or vertex is coupled with 15 other ones) [21]. This enhanced connectivity allows for better utilization of the existing qubits, thus fewer vertices are capable of broader calculations.

Table 1. Evolution of Chimera graphs throughout previous D-Wave versions (from [21]).

	Size of quantum processing unit	Total number of qubits
D-Wave One	4×4	128
D-Wave Two	8×8	512
D-Wave 2X	12×12	1152
D-Wave 2000Q	16×16	2048

Lucas in [32] discussed Ising formulations for a variety of NP-complete and NP-hard optimization problems (including the TSP problem), with emphasis on using as few as possible qubits. Martoňák et al. in [34] introduced a different quantum annealing scheme based on path-integral Monte Carlo processes to address the symmetric version of the Traveling Salesman Problem (sTSP). Their approach is built upon a rather constrained Ising-like representation and is compared against the standard simulated annealing heuristic on various benchmark tests, demonstrating its superiority.

Boros et al. presented a set of local search heuristics for Quadratic Unconstrained Binary Optimization (QUBO) problems, providing indicative simulation results on various benchmark tests [29]. Adiabatic quantum annealing techniques are also used to address multiobjective optimization problems. In particular, Barán and Villagra proved their algorithm is capable of finding Pareto-optimal solutions in finite-time for a particular class of problems [60].

Another work on quantum annealing and TSP was presented by Warren in [36]. Warren studied small-scale instances of traveling salesman problems, showcasing how a D-Wave machine using quantum annealing would operate to solve these instance. The motivation for this work was to offer a tutorial-like approach, since the limitations on the number of TSP nodes are quite restrictive for real-world applications. The mapping of the CMO protein problem to a QUBO formulation was studied by Oliveira et al. in [61]. Simulation results showed that the proposed approach outperformed classical techniques.

On the other hand, Ushijima-Mwesigwa demonstrated the graph partitioning mechanism of D-Wave computers utilizing the quantum annealing tools on the D-Wave 2X [30]. The reduction of the large matrix size in QUBO was the main topic in [24] by Lewis and Glover. This was vital in order to have a quick solution for large-scale problems with numerous variables that require equally many qubits. Glover et al. showed in a step by step procedure how one can translate a problem with particular characteristics into a QUBO instance, using the appropriate tools [28]. Another iterative version of the quantum annealing heuristic for QUBO problems based on tabu search was presented by Rosenberg et al. in [62]. Regarding the technical details of that work, their approach tries to partition the problem into subproblems, while keeping the rest of variables fixed. Moreover, they consider the effect of the time to reach the best solution on the problems size.

Many researchers have studied the classical Traveling Salesman Problem with Time Windows (TSPTW). The literature can provide exact algorithms for solving the TSPTW. Langevin et al. [63] introduced a flow formulation of two elements, which can be extended to the classic "makespan" problem. Dumas et al. [64] used a dynamic programming approach reducing trials, which improved the performance and scaled down the search space, in advance and during its execution as well. The TSPTW is an NP-hard problem, since it is a special case of the famous TSP. So, in practice, a heuristic, which is able to solve effectively realistic cases within a reasonable time, is necessary. Gendreau et al. [65] proposed an insertion heuristic, which gradually builds the path to the construction phase and improves a refinement phase. Urrutia et al. [66] proposed a two-stage heuristic, based on VNS. In the first step, a feasible solution is manufactured using the VNS, wherein the mixed linear, integer, objective function is represented as an infeasibility measurement. In the second stage, the heuristic improves the feasible solution with a general version of VNS (General VNS - GVNS).

A hybrid genetic algorithm for finding guaranteed and reliable solutions of global optimization problems using the branch-and-bound technique was proposed by Sotiropoulos et al. [67]. A branch-and-bound approach was, also, used in the work of Pardalos et al. in [68], where dynamic preprocessing techniques and heuristics are used to calculate good initial configurations. For an in-depth insight on quantum genetic algorithms, the reader is referred to the work of Lahoz-Beltra in [69]).

Papalitsas et al. in [70] developed a metaheuristic based on conventional principles for finding within a short period of time feasible solutions for the TSPTW. Subsequently, a novel quantum-inspired unconventional metaheuristic method, based on the original General Variable Neighborhood Search (GVNS), was proposed in order to solve the standard TSP [71]. This quantum inspired metaheuristic was applied to the solution of the garbage collection problem modeled as a TSP instance [17]. Recently, Papalitsas et al. [72] applied this quantum-inspired metaheuristic to the practical real-life problem of garbage collection with time windows. For the majority of the benchmark instances used to evaluate the proposed metaheuristic, the experimental results were particularly promising. Towards the ultimate goal of running the TSPTW using pure quantum optimization methods, we focus here on its QUBO formulation. This present article is an attempt in that direction.

3. The classical formulation of the TSPTW

In this section we give the formal definition of the classical TSPTW. Our presentation follows [73], which is pretty much standard in the relative literature.

Definition 1. *Let $G = (N, A)$ be a directed graph, where $N = \{0, 1, \ldots, n\}$ is the finite set of nodes or, more commonly referred to in this context as* customers, *and $A = N \times N$ is the set of arcs connecting the customers. For every pair (u, v) of customers there exists an arc in A. A* tour *is defined by the order in which the customers are visited.*

A couple of assumptions facilitate the formulation of the TSPTW.

Definition 2. *Let customer 0 denote the depot and assume that every tour* begins *and* ends *at the depot. Each of the remaining n customers appears exactly once in the tour. We denote a tour as an ordered list $\mathcal{P} = (p_0, p_1, \ldots, p_n, p_{n+1})$, where p_i is the index of the customer in the i^{th} position of the tour. According to our previous assumption $p_0 = p_{n+1} = 0$.*

Definition 3. *For every pair (u, v) of customers $u, v \in N$, there is a cost $c_{u,v}$, for traversing the arc (u, v). This cost of traversing the arc from u to v generally consists of any* service time *at customer U plus the* travel time *from customer u to customer v.*

Definition 4. *To each customer $v \in N$ there is an associated* time window *$[e_v, l_v]$, during which the customer v must be visited. We assume that waiting is permitted; a vehicle is allowed to reach customer v before the beginning of its time window, e_v, but the vehicle cannot depart from customer v before e_v.*

A tour is feasible *if it satisfies the time window of each customer.*

In the literature two primary TSPTW objective functions are usually considered

- minimize the sum of the arc traversal costs along the tour, and
- minimize the time to return to the depot.

In a way, the difficulty of the TSPTW stems from the fact that it is two problems in one; a traveling salesman problem and a scheduling problem. The TSP alone is one of the most famous NP-hard optimization problems, while the scheduling part, with release and due dates, adds considerably to the already existing difficulty. To verify that the tour is feasible, i.e., it satisfies the time windows, it is expedient to introduce the **arrival time** at the i^{th} customer and the time at which **service** starts at the i^{th} customer, which are denoted by A_{p_i} and D_{p_i}, respectively. At this point we make the important observation that D_{p_i} is the **departure time** from the i^{th} customer in the case of **zero service time**. The assumption of zero service time is widely used in the literature in order to simplify the problem, and, so, we too shall follow this assumption in our presentation.

The classical formulation of the TSPTW can be summarized by the next relations (see also [73]).

$$\min \sum_{i=1}^{n+1} c_{p_{i-1},p_i} \tag{1}$$

In the above expression (1), we assume that $(p_0, p_1, \ldots, p_n, p_{n+1})$ is a feasible tour. This means that, besides the assumptions previously outlined, the following hold.

$$D_{p_0} = 0\,. \tag{2}$$

$$A_{p_i} = D_{p_{i-1}} + c_{p_{i-1},p_i} \quad (1 \le i \le n+1)\,. \tag{3}$$

$$D_{p_i} = \max\{A_{p_i}, e_{p_i}\} \quad (1 \le i \le n)\,. \tag{4}$$

3.1. An illustrated example for the TSPTW

At this subsection we shall describe and explain a template reference problem with 4 nodes plus the starting point (5 nodes in total). Although this example is quite simple, we hope that it will help the reader to easily understand the TSPTW. Specifically, to better comprehend the modeling of this benchmark, as well as the attempt to find a feasible solution at first, and consequently the optimal one. The next Figure 1 is the graphical depiction of the 5 customers and all arcs that connect them.

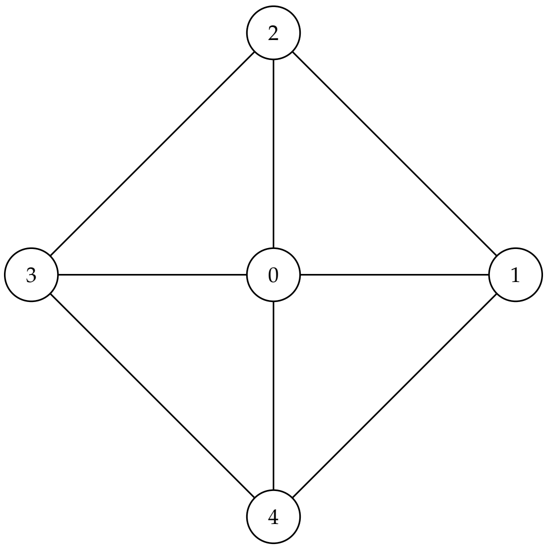

Figure 1. The above graph depicts an example of a tour consisting of 4 nodes plus the depot (node 0).

Table 2 includes all the relevant data of the above example. In particular, we see the coordinates X and Y of every node, as well as the Ready Time and the Due Date.

Table 2. The input data for our example.

Node No	X	Y	Ready Time	Due Date
1	2	2	1	30
2	2	3	14	15
3	3	3	12	25
4	3	4	4	5
5	2	1	8	10

Table 3 is actually the distance matrix for our example. Let us clarify here that we calculated the costs between the nodes using the Euclidean distance, which is given by the well-know formula: $d(u,v) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, where $u = (x_1, y_1)$ and $v = (x_2, y_2)$.

Table 3. The distance matrix.

	Node 1	Node 2	Node 3	Node 4	Node 5
Node 1	0	1	1.41	2.23	1
Node 2	1	0	1	1.41	2
Node3	1.41	1	0	1	2.23
Node 4	2.23	1.41	1	0	3.16
Node 5	1	2	2.23	3.16	0

A feasible solution for this particular TSPTW consisting of 5 nodes is given in Table 4. One can see that if all time windows are respected in every step of the tour going from from one customer to the next, a feasible solution can be constructed. Although, this is a small scale example, we expect that it will enhance one's understanding of what is a feasible solution for the TSPTW.

Table 4. A feasible solution of our illustrated example.

Ordering	Node 1	Node 4	Node 5	Node 3	Node 2
cost	0<1	2.23 <4	7.16 <8	10 <12	12<14
feasibility	yes	yes	yes	yes	yes

In the next section we introduce our novel approach for mapping the TSPTW over the quadratic unconstrained binary optimization (QUBO) model.

4. A QUBO formulation for the TSPTW

In the literature, the typical QUBO formulation of the standard TSP involves the use of a set of *binary* variables (see [32]). They are characterized as binary in the sense that they can take only the 0 or 1 value. Typically, they are denoted by $x_{v,i}$, and their meaning is the following:

$$x_{v,i} = \begin{cases} 1, & \text{customer } v \text{ is at position } i \text{ in the tour} \\ 0, & \text{otherwise} \end{cases} . \tag{5}$$

In the case of the TSPTW, we have discovered to be more advantageous to use binary variables parameterized by three integers: u, v and i. A similar suggestion for the Vehicle Routing Problem can be found in [38]. Hence, in the rest of our analysis, we shall use the binary variables $x^i_{u,v}$ defined next:

$$x^i_{u,v} = \begin{cases} 1, & \text{customers } u \text{ and } v \text{ are at consecutive positions } i-1 \text{ and } i \text{ in the tour} \\ 0, & \text{otherwise} \end{cases} . \tag{6}$$

As we explained in the previous section, a feasible tour has the form $\{p_0, p_1, \ldots, p_n, p_{n+1}\}$. In this enumeration, p_i is the customer in the i^{th} position of the tour. We always take for granted that $p_0 = p_{n+1} = 0$ and each other customer appears exactly once. We recall the assumption of *zero service time* at each customer, and without loss of generality, we adopt another assumption that greatly reduces the final clutter of our QUBO expressions: for each customer v, where $0 \leq v \leq n+1$, $e_v = 0$. With this understanding, we see that the parameters u and v range from 0 to n and the parameter i ranges from 1 to $n+1$.

Therefore, we can assert that:

- For each i, $1 \leq i \leq n+1$, exactly one of the binary variables $x^i_{u,v}$ is 1, where u and v range freely from 0 to n, with the proviso that $u \neq v$. As a matter of fact when $i = 1$ and $i = n+1$ we can be more precise. In the former case, exactly one of the binary variables $x^1_{0,v}$ is 1, where v ranges from 1 to n, and, in the latter case, exactly one of the binary variables $x^{n+1}_{v,0}$ is 1, where v ranges from 1 to n.

- In addition to the above constraints, for each u, $1 \leq u \leq n$, exactly one of the binary variables $x^i_{u,v}$ is 1, where i ranges from 2 to n and v ranges from 1 to n.
- Symmetrically, we also have the constraint that for every v, $1 \leq v \leq n$, exactly one of the binary variables $x^i_{u,v}$ is 1, where i ranges from 2 to n and u ranges from 1 to n.

These constraints are encoded in the Hamiltonian H_c.

$$\begin{aligned} H_c = B \sum_{i=1}^{n+1} \left(1 - \sum_{u=0}^{n} \sum_{\substack{v=0 \\ v \neq u}}^{n} x^i_{u,v} \right)^2 + B \left(1 - \sum_{v=1}^{n} x^1_{0,v} \right)^2 + B \left(1 - \sum_{v=1}^{n} x^{n+1}_{v,0} \right)^2 \\ + B \sum_{u=1}^{n} \left(1 - \sum_{i=2}^{n} \sum_{v=1}^{n} x^i_{u,v} \right)^2 \\ + B \sum_{v=1}^{n} \left(1 - \sum_{i=2}^{n} \sum_{u=1}^{n} x^i_{u,v} \right)^2 \end{aligned} \tag{7}$$

Using the $x^i_{u,v}$ binary variables, the requirement that the tour be minimal can be encoded by the following Hamiltonian H_m.

$$H_m = C \sum_{i=1}^{n+1} \sum_{u=0}^{n} \sum_{\substack{v=0 \\ v \neq u}}^{n} x^i_{u,v} c_{u,v} \; . \tag{8}$$

In the above Hamiltonians, B and C are positive constants, which must be appropriately chosen, i.e., $C < B$, so as to ensure that the constraints of H_c are respected (see also [32]). Obviously, $c_{u,v}$ is the cost for traversing the arc (u, v).

The above Hamiltonians are certainly not the end of the story in the case of the TSPTW, as, in their essence, they just encode the minimal cost of the Hamiltonian circuit. There are a lot more difficult time constraints to tackle in order to satisfy the time window of every customer. To this end, besides the binary variables $x^i_{u,v}$, it will be necessary to use a second type of binary variables, denoted by $t_{n,i}$, in order to express the *time margin* of every customer.

Definition 5. *Given a* feasible *tour* $p_0, p_1, \ldots, p_n, p_{n+1}$*, suppose that the customer at position* i*, where* $1 \leq i \leq n$*, is* v *with time window* $[e_v, l_v]$*. We say that the* ***time margin*** *of the customer at position* i *to be* $l_v - A_{p_i}$.

Clearly, for a feasible tour, the time margin for every position of the tour is non negative. We can now define the binary variables $t_{k,i}$ as follows:

$$t_{k,i} = \begin{cases} 1, & \text{the time margin of the customer at position } i \text{ in the tour is } k \\ 0, & \text{otherwise} \end{cases} . \tag{9}$$

We recall that in the formulation of the TSPTW, the arrival time at the customer in the i^{th} position of the tour, denoted A_{p_i}, plays an important role (see [73]). Thus, we begin our analysis by showing how to express A_{p_i}. Obviously, $A_{p_0} = 0$, so it is only necessary to give the formula for A_{p_i}, $1 \leq i \leq n$. We first observe that the customer at position 0 is always the depot (customer 0), which results in the following, relatively simple formula, for A_{p_1}.

$$A_{p_1} = \sum_{v=1}^{n} x^1_{0,v} c_{0,v} \; . \tag{10}$$

The general case, i.e., when $2 \leq i \leq n$, is taken care by the next equation.

$$A_{p_i} = \sum_{d=1}^{i} \sum_{u=1}^{n} \sum_{\substack{v=1 \\ v \neq u}}^{n} x^d_{u,v} c_{u,v} \quad (2 \leq i \leq n) \,. \tag{11}$$

Example 1. *To explain how Equations (10) and (11) can be used in practice, we continue with our test case example.*

Equation (10) becomes

$$A_{p_1} = x^1_{0,1}c_{0,1} + x^1_{0,2}c_{0,2} + x^1_{0,3}c_{0,3} + x^1_{0,4}c_{0,4} \tag{12}$$

Equation (11) gives the following series of equations.

$$\begin{aligned} A_{p_2} = &\left(\underbrace{x^1_{0,1}c_{0,1} + x^1_{0,2}c_{0,2} + x^1_{0,3}c_{0,3} + x^1_{0,4}c_{0,4}}_{A_{p_1}} \right) \\ &+ x^2_{1,2}c_{1,2} + x^2_{1,3}c_{1,3} + x^2_{1,4}c_{1,4} + x^2_{2,1}c_{2,1} + x^2_{2,3}c_{2,3} + x^2_{2,4}c_{2,4} \\ &+ x^2_{3,1}c_{3,1} + x^2_{3,2}c_{3,2} + x^2_{3,4}c_{3,4} + x^2_{4,1}c_{4,1} + x^2_{4,2}c_{4,2} + x^2_{4,3}c_{4,3} \end{aligned} \tag{13}$$

$$\begin{aligned} A_{p_3} = &\Bigg(\left(x^1_{0,1}c_{0,1} + x^1_{0,2}c_{0,2} + x^1_{0,3}c_{0,3} + x^1_{0,4}c_{0,4} \right) \\ &+ x^2_{1,2}c_{1,2} + x^2_{1,3}c_{1,3} + x^2_{1,4}c_{1,4} + x^2_{2,1}c_{2,1} + x^2_{2,3}c_{2,3} + x^2_{2,4}c_{2,4} \\ &+ x^2_{3,1}c_{3,1} + x^2_{3,2}c_{3,2} + x^2_{3,4}c_{3,4} + x^2_{4,1}c_{4,1} + x^2_{4,2}c_{4,2} + x^2_{4,3}c_{4,3} \Bigg) \\ &+ x^3_{1,2}c_{1,2} + x^3_{1,3}c_{1,3} + x^3_{1,4}c_{1,4} + x^3_{2,1}c_{2,1} + x^3_{2,3}c_{2,3} + x^3_{2,4}c_{2,4} \\ &+ x^3_{3,1}c_{3,1} + x^3_{3,2}c_{3,2} + x^3_{3,4}c_{3,4} + x^3_{4,1}c_{4,1} + x^3_{4,2}c_{4,2} + x^3_{4,3}c_{4,3} \end{aligned} \tag{14}$$

$$\begin{aligned} A_{p_4} = &\, x^1_{0,1}c_{0,1} + x^1_{0,2}c_{0,2} + x^1_{0,3}c_{0,3} + x^1_{0,4}c_{0,4} \\ &+ x^2_{1,2}c_{1,2} + x^2_{1,3}c_{1,3} + x^2_{1,4}c_{1,4} + x^2_{2,1}c_{2,1} + x^2_{2,3}c_{2,3} + x^2_{2,4}c_{2,4} \\ &+ x^2_{3,1}c_{3,1} + x^2_{3,2}c_{3,2} + x^2_{3,4}c_{3,4} + x^2_{4,1}c_{4,1} + x^2_{4,2}c_{4,2} + x^2_{4,3}c_{4,3} \\ &+ x^3_{1,2}c_{1,2} + x^3_{1,3}c_{1,3} + x^3_{1,4}c_{1,4} + x^3_{2,1}c_{2,1} + x^3_{2,3}c_{2,3} + x^3_{2,4}c_{2,4} \\ &+ x^3_{3,1}c_{3,1} + x^3_{3,2}c_{3,2} + x^3_{3,4}c_{3,4} + x^3_{4,1}c_{4,1} + x^3_{4,2}c_{4,2} + x^3_{4,3}c_{4,3} \\ &+ x^4_{1,2}c_{1,2} + x^4_{1,3}c_{1,3} + x^4_{1,4}c_{1,4} + x^4_{2,1}c_{2,1} + x^4_{2,3}c_{2,3} + x^4_{2,4}c_{2,4} \\ &+ x^4_{3,1}c_{3,1} + x^4_{3,2}c_{3,2} + x^4_{3,4}c_{3,4} + x^4_{4,1}c_{4,1} + x^4_{4,2}c_{4,2} + x^4_{4,3}c_{4,3} \end{aligned} \tag{15}$$

The above equations demonstrate that in every case, A_{p_i} can be expressed as a sum of terms, where each term is the product of an input variable $c_{u,v}$ and exactly one *binary decision variable $x^i_{u,v}$.*

The simplifying assumption of zero service time enables us to express the constraints imposed by the time windows of every customer as follows:

$$A_{p_1} \leq \sum_{v=1}^{n} x^1_{0,v} l_v \,, \tag{16}$$

for the special case where $i = 1$, and

$$A_{p_i} \leq \sum_{u=1}^{n} \sum_{\substack{v=1 \\ v \neq u}}^{n} x^i_{u,v} l_v \quad (2 \leq i \leq n) \,, \tag{17}$$

for the general case where $2 \leq i \leq n$.

The above expression may seem a little complicated, but, unfortunately, while A_{p_i} tells us the arrival time at the customer in the i^{th} position of the tour, it does not tell us *which is* this particular customer. We have to resort to the binary variables $x^i_{u,v}$ to indirectly obtain this information.

Inequality constraints such as these in (16) and (17) are notoriously difficult to express within the QUBO framework. For an extensive analysis we refer the interested reader to [28,59,74]. The approach which is most commonly used in the literature is to employ auxiliary binary variables, like the binary variables $t_{k,i}$ previously defined, to convert the inequality into an equality, and then proceed, as usual, by squaring the equality constraint.

In our case, the first step is to express the inequalities (16) and (17) as

$$A_{p_1} + \sum_{k=1}^{K} k t_{k,1} = \sum_{v=1}^{n} x^1_{0,v} l_v \,, \tag{18}$$

and as

$$A_{p_i} + \sum_{k=1}^{K} k t_{k,i} = \sum_{u=1}^{n} \sum_{\substack{v=1 \\ v \neq u}}^{n} x^i_{u,v} l_v \quad (2 \leq i \leq n) \,, \tag{19}$$

respectively.

In the above equalities K is a positive constant appropriately chosen taking into consideration the specific time windows. A valid possible choice could be $K = \max_{1 \leq v \leq n}\{l_v\}$. Such a choice, while valid, would be unnecessarily big in most practical cases.

Equality constraints like (18) and (19) are typically treated in QUBO by converting them into squared expressions. Hence, (18) gives rise to the first time window constraint, denoted by W_1 and given by

$$W_1 = \left(A_{p_1} + \sum_{k=1}^{K} k t_{k,i} - \sum_{v=1}^{n} x^1_{0,v} l_v \right)^2 \,, \tag{20}$$

while Equation (19) gives rise to the i^{th} time window constraint, denoted by W_i and given by:

$$W_i = \left(A_{p_i} + \sum_{k=1}^{K} k t_{k,i} - \sum_{u=1}^{n} \sum_{\substack{v=1 \\ v \neq u}}^{n} x^i_{u,v} l_v \right)^2 \quad (2 \leq i \leq n) \,. \tag{21}$$

If we replace A_{p_1} and A_{p_i} in the above equations by the formulas in (10) and (11), we derive the expanded forms of W_1 and W_i, $2 \leq i \leq n$, respectively.

$$W_1 = \left(\sum_{v=1}^{n} x^1_{0,v} c_{0,v} + \sum_{k=1}^{K} k t_{k,i} - \sum_{v=1}^{n} x^1_{0,v} l_v \right)^2 \,. \tag{22}$$

$$W_i = \left(\sum_{d=1}^{i} \sum_{u=1}^{n} \sum_{\substack{v=1 \\ v \neq u}}^{n} x_{u,v}^{d} c_{u,v} + \sum_{k=1}^{K} k t_{k,i} - \sum_{u=1}^{n} \sum_{\substack{v=1 \\ v \neq u}}^{n} x_{u,v}^{i} l_v \right)^2 \quad (2 \leq i \leq n) . \tag{23}$$

At this point it is important to pause and confirm that the constraints in Equations (22) and (23) conform to the QUBO formulation requirements, in the sense that, after the expansion of the square, we get a sum of terms, where each term is the product of input data like $c_{u,v}$ or l_v and *at most two* binary decision variables.

The last time constraint concerns the binary variables $t_{k,i}$. For each i, $1 \leq i \leq n$, exactly one of the binary variables $t_{k,i}$ is 1, where k ranges from 1 to K. The meaning of this constraint is obvious: in every position of a feasible tour the time margin should be unique. Expressing this constraint is also straightforward:

$$M_i = \left(1 - \sum_{k=1}^{K} t_{k,i} \right)^2 \quad (1 \leq i \leq n) . \tag{24}$$

Putting all the time constraints together results in the Hamiltonian H_t:

$$H_t = T(W_1) + T \sum_{i=2}^{n} TWC_i + T \sum_{i=1}^{n} M_i . \tag{25}$$

Therefore, to solve the TSPTW in the QUBO framework we must use the Hamiltonian H given below:

$$H = H_c + H_m + H_t . \tag{26}$$

As noted earlier, the constants B, C and T appearing in the Hamiltonians are positive constants, which must be chosen according to our requirements. For instance, by setting $C < B$, so as to ensure that the constraints of H_c are respected; similarly, setting $T < B$ prioritizes the time windows constraints over the minimality of the tour.

Example 2. *To show the form of the time windows constraints when the square is expanded, we apply constraint (22) to our test case example.*

First we point out that for binary variables the following hold:

$$(x_{u,v}^{i})^2 = x_{u,v}^{i} \quad \textit{and} \quad (t_{k,i})^2 = t_{k,i} . \tag{27}$$

We also recall the identity: $(a + b - c)^2 = a^2 + b^2 + c^2 + 2ab - 2ac - 2bc$. We shall use this identity to expand (22), setting $a = \sum_{v=1}^{n} x_{0,v}^{1} c_{0,v}$, $b = \sum_{k=1}^{K} k t_{k,1}$, and $c = \sum_{v=1}^{n} x_{0,v}^{1} l_v$. In our example $n = 4$ and, in order to simplify somewhat the calculations we take $K = 2$. With this understanding we use Equation (12) to derive the formulas given below. Note that to improve the readability of the equations in this example, we have written in green color those terms that involve exactly one *binary variable and in blue color those terms that involve* exactly two *binary variables.*

$$\begin{aligned}A_{p_1}^2 &= \left(x_{0,1}^1 c_{0,1} + x_{0,2}^1 c_{0,2} + x_{0,3}^1 c_{0,3} + x_{0,4}^1 c_{0,4}\right)^2 \\ &= (x_{0,1}^1)^2 c_{0,1}^2 + (x_{0,2}^1)^2 c_{0,2}^2 + (x_{0,3}^1)^2 c_{0,3}^2 + (x_{0,4}^1)^2 c_{0,4}^2 + 2x_{0,1}^1 c_{0,1} x_{0,2}^1 c_{0,2} + 2x_{0,1}^1 c_{0,1} x_{0,3}^1 c_{0,3} \\ &+ 2x_{0,1}^1 c_{0,1} x_{0,4}^1 c_{0,4} + 2x_{0,2}^1 c_{0,2} x_{0,3}^1 c_{0,3} + 2x_{0,2}^1 c_{0,2} x_{0,4}^1 c_{0,4} + 2x_{0,3}^1 c_{0,3} x_{0,4}^1 c_{0,4} \\ &= x_{0,1}^1 c_{0,1}^2 + x_{0,2}^1 c_{0,2}^2 + x_{0,3}^1 c_{0,3}^2 + x_{0,4}^1 c_{0,4}^2 + 2x_{0,1}^1 c_{0,1} x_{0,2}^1 c_{0,2} + 2x_{0,1}^1 c_{0,1} x_{0,3}^1 c_{0,3} \\ &+ 2x_{0,1}^1 c_{0,1} x_{0,4}^1 c_{0,4} + 2x_{0,2}^1 c_{0,2} x_{0,3}^1 c_{0,3} + 2x_{0,2}^1 c_{0,2} x_{0,4}^1 c_{0,4} + 2x_{0,3}^1 c_{0,3} x_{0,4}^1 c_{0,4}\end{aligned} \tag{28}$$

376 *Similarly, we see that:*

$$\left(\sum_{k=1}^{2} k t_{k,1}\right)^2 = (1t_{1,1} + 2t_{2,1})^2 = t_{1,1}^2 + 4t_{2,1}^2 + 4t_{1,1}t_{2,1} = t_{1,1} + 4t_{2,1} + 4t_{1,1}t_{2,1} \tag{29}$$

$$\begin{aligned}\left(\sum_{v=1}^{4} x_{0,v}^1 l_v\right)^2 &= \left(x_{0,1}^1 l_1 + x_{0,2}^1 l_2 + x_{0,3}^1 l_3 + x_{0,4}^1 l_4\right)^2 \\ &= (x_{0,1}^1)^2 l_1^2 + (x_{0,2}^1)^2 l_2^2 + (x_{0,3}^1)^2 l_3^2 + (x_{0,4}^1)^2 l_4^2 + 2x_{0,1}^1 l_1 x_{0,2}^1 l_2 + 2x_{0,1}^1 l_1 x_{0,3}^1 l_3 \\ &+ 2x_{0,1}^1 l_1 x_{0,4}^1 l_4 + 2x_{0,2}^1 l_2 x_{0,3}^1 l_3 + 2x_{0,2}^1 l_2 x_{0,4}^1 l_4 + 2x_{0,3}^1 l_3 x_{0,4}^1 l_4 \\ &= x_{0,1}^1 l_1^2 + x_{0,2}^1 l_2^2 + x_{0,3}^1 l_3^2 + x_{0,4}^1 l_4^2 + 2x_{0,1}^1 l_1 x_{0,2}^1 l_2 + 2x_{0,1}^1 l_1 x_{0,3}^1 l_3 \\ &+ 2x_{0,1}^1 l_1 x_{0,4}^1 l_4 + 2x_{0,2}^1 l_2 x_{0,3}^1 l_3 + 2x_{0,2}^1 l_2 x_{0,4}^1 l_4 + 2x_{0,3}^1 l_3 x_{0,4}^1 l_4\end{aligned} \tag{30}$$

$$\begin{aligned}2A_{p_1} \sum_{k=1}^{2} k t_{k,1} &= 2\left(x_{0,1}^1 c_{0,1} + x_{0,2}^1 c_{0,2} + x_{0,3}^1 c_{0,3} + x_{0,4}^1 c_{0,4}\right)(1t_{1,1} + 2t_{2,1}) \\ &= 2x_{0,1}^1 c_{0,1} t_{1,1} + 2x_{0,1}^1 c_{0,1} t_{2,1} + 2x_{0,2}^1 c_{0,2} t_{1,1} + 2x_{0,2}^1 c_{0,2} t_{2,1} \\ &+ 2x_{0,3}^1 c_{0,3} t_{1,1} + 2x_{0,3}^1 c_{0,3} t_{2,1} + 2x_{0,4}^1 c_{0,4} t_{1,1} + 2x_{0,4}^1 c_{0,4} t_{2,1}\end{aligned} \tag{31}$$

$$\begin{aligned}2A_{p_1} \sum_{v=1}^{4} x_{0,v}^1 l_v &= 2\left(x_{0,1}^1 c_{0,1} + x_{0,2}^1 c_{0,2} + x_{0,3}^1 c_{0,3} + x_{0,4}^1 c_{0,4}\right)\left(x_{0,1}^1 l_1 + x_{0,2}^1 l_2 + x_{0,3}^1 l_3 + x_{0,4}^1 l_4\right) \\ &= 2(x_{0,1}^1)^2 c_{0,1} l_1 + 2x_{0,1}^1 c_{0,1} x_{0,2}^1 l_2 + 2x_{0,1}^1 c_{0,1} x_{0,3}^1 l_3 + 2x_{0,1}^1 c_{0,1} x_{0,4}^1 l_4 \\ &+ 2x_{0,2}^1 c_{0,2} x_{0,1}^1 l_1 + 2(x_{0,2}^1)^2 c_{0,2} l_2 + 2x_{0,2}^1 c_{0,2} x_{0,3}^1 l_3 + 2x_{0,2}^1 c_{0,2} x_{0,4}^1 l_4 \\ &+ 2x_{0,3}^1 c_{0,3} x_{0,1}^1 l_1 + 2x_{0,3}^1 c_{0,3} x_{0,2}^1 l_2 + 2(x_{0,3}^1)^2 c_{0,3} l_3 + 2x_{0,3}^1 c_{0,3} x_{0,4}^1 l_4 \\ &+ 2x_{0,4}^1 c_{0,4} x_{0,1}^1 l_1 + 2x_{0,4}^1 c_{0,4} x_{0,2}^1 l_2 + 2x_{0,4}^1 c_{0,4} x_{0,3}^1 l_3 + 2(x_{0,4}^1)^2 c_{0,4} l_4 \\ &= 2x_{0,1}^1 c_{0,1} l_1 + 2x_{0,2}^1 c_{0,2} l_2 + 2x_{0,3}^1 c_{0,3} l_3 + 2x_{0,4}^1 c_{0,4} l_4 \\ &+ 2x_{0,1}^1 c_{0,1} x_{0,2}^1 l_2 + 2x_{0,1}^1 c_{0,1} x_{0,3}^1 l_3 + 2x_{0,1}^1 c_{0,1} x_{0,4}^1 l_4 \\ &+ 2x_{0,2}^1 c_{0,2} x_{0,1}^1 l_1 + 2x_{0,2}^1 c_{0,2} x_{0,3}^1 l_3 + 2x_{0,2}^1 c_{0,2} x_{0,4}^1 l_4 \\ &+ 2x_{0,3}^1 c_{0,3} x_{0,1}^1 l_1 + 2x_{0,3}^1 c_{0,3} x_{0,2}^1 l_2 + 2x_{0,3}^1 c_{0,3} x_{0,4}^1 l_4 \\ &+ 2x_{0,4}^1 c_{0,4} x_{0,1}^1 l_1 + 2x_{0,4}^1 c_{0,4} x_{0,2}^1 l_2 + 2x_{0,4}^1 c_{0,4} x_{0,3}^1 l_3\end{aligned} \tag{32}$$

$$
\begin{aligned}
2\sum_{k=1}^{2} k t_{k,1} \sum_{v=1}^{4} x^1_{0,v} l_v &= 2\left(1t_{1,1} + 2t_{2,1}\right)\left(x^1_{0,1} l_1 + x^1_{0,2} l_2 + x^1_{0,3} l_3 + x^1_{0,4} l_4\right) \\
&= 2t_{1,1}x^1_{0,1}l_1 + 2t_{1,1}x^1_{0,2}l_2 + 2t_{1,1}x^1_{0,3}l_3 + 2t_{1,1}x^1_{0,4}l_4 \\
&+ 4t_{2,1}x^1_{0,1}l_1 + 4t_{2,1}x^1_{0,2}l_2 + 4t_{2,1}x^1_{0,3}l_3 + 4t_{2,1}x^1_{0,4}l_4
\end{aligned} \tag{33}
$$

We can now substitute Equations (28)–(33) in (22) to finally arrive at the expanded form of the constraint, given by the following equation.

$$
\begin{aligned}
W_1 = {} & x^1_{0,1}c^2_{0,1} + x^1_{0,2}c^2_{0,2} + x^1_{0,3}c^2_{0,3} + x^1_{0,4}c^2_{0,4} + 2x^1_{0,1}c_{0,1}x^1_{0,2}c_{0,2} + 2x^1_{0,1}c_{0,1}x^1_{0,3}c_{0,3} \\
& + 2x^1_{0,1}c_{0,1}x^1_{0,4}c_{0,4} + 2x^1_{0,2}c_{0,2}x^1_{0,3}c_{0,3} + 2x^1_{0,2}c_{0,2}x^1_{0,4}c_{0,4} + 2x^1_{0,3}c_{0,3}x^1_{0,4}c_{0,4} \\
& + t_{1,1} + 4t_{2,1} + 4t_{1,1}t_{2,1} \\
& + x^1_{0,1}l^2_1 + x^1_{0,2}l^2_2 + x^1_{0,3}l^2_3 + x^1_{0,4}l^2_4 + 2x^1_{0,1}l_1x^1_{0,2}l_2 + 2x^1_{0,1}l_1x^1_{0,3}l_3 \\
& + 2x^1_{0,1}l_1x^1_{0,4}l_4 + 2x^1_{0,2}l_2x^1_{0,3}l_3 + 2x^1_{0,2}l_2x^1_{0,4}l_4 + 2x^1_{0,3}l_3x^1_{0,4}l_4 \\
& - 2x^1_{0,1}c_{0,1}t_{1,1} - 2x^1_{0,1}c_{0,1}t_{2,1} - 2x^1_{0,2}c_{0,2}t_{1,1} - 2x^1_{0,2}c_{0,2}t_{2,1} \\
& - 2x^1_{0,3}c_{0,3}t_{1,1} - 2x^1_{0,3}c_{0,3}t_{2,1} - 2x^1_{0,4}c_{0,4}t_{1,1} - 2x^1_{0,4}c_{0,4}t_{2,1} \\
& - 2x^1_{0,1}c_{0,1}l_1 - 2x^1_{0,2}c_{0,2}l_2 - 2x^1_{0,3}c_{0,3}l_3 - 2x^1_{0,4}c_{0,4}l_4 \\
& - 2x^1_{0,1}c_{0,1}x^1_{0,2}l_2 - 2x^1_{0,1}c_{0,1}x^1_{0,3}l_3 - 2x^1_{0,1}c_{0,1}x^1_{0,4}l_4 \\
& - 2x^1_{0,2}c_{0,2}x^1_{0,1}l_1 - 2x^1_{0,2}c_{0,2}x^1_{0,3}l_3 - 2x^1_{0,2}c_{0,2}x^1_{0,4}l_4 \\
& - 2x^1_{0,3}c_{0,3}x^1_{0,1}l_1 - 2x^1_{0,3}c_{0,3}x^1_{0,2}l_2 - 2x^1_{0,3}c_{0,3}x^1_{0,4}l_4 \\
& - 2x^1_{0,4}c_{0,4}x^1_{0,1}l_1 - 2x^1_{0,4}c_{0,4}x^1_{0,2}l_2 - 2x^1_{0,4}c_{0,4}x^1_{0,3}l_3 \\
& - 2t_{1,1}x^1_{0,1}l_1 - 2t_{1,1}x^1_{0,2}l_2 - 2t_{1,1}x^1_{0,3}l_3 - 2t_{1,1}x^1_{0,4}l_4 \\
& - 4t_{2,1}x^1_{0,1}l_1 - 4t_{2,1}x^1_{0,2}l_2 - 4t_{2,1}x^1_{0,3}l_3 - 4t_{2,1}x^1_{0,4}l_4
\end{aligned} \tag{34}
$$

Similar calculations, albeit too lengthy to include here, confirm that all time window constraints have similar patterns, that is they constitute legitimate expressions within the QUBO framework.

5. Conclusions and future work

In this work, we have considered the TSPTW. Although many combinatorial optimization problems have been expressed in the QUBO (or, equivalently, the Ising) formulation, this particular problem was not one of them. Presumably, the reason is the TSPTW imposes many additional (time) constraints because the customers' time windows must be satisfied. These are actually inequality constraints that are very difficult to tackle within the QUBO framework. We remind the reader that valid QUBO expressions must have the form: $x^T Q x$, where x is a column vector of binary decision variables, x^T its transpose and Q a square matrix of constants. So, to the best of our knowledge, this is the first time the TSPTW is cast in QUBO form.

This step is a necessary precondition in order to be able to run TSPTW instances on the current generation of D-Wave computers. Hence, the future direction of this work will the mapping of TSPTW benchmarks to the Chimera or the upcoming Pegasus architecture, so as to obtain experimental results. This will enable the comparison of the current state of the art classical, or conventional, if you prefer, metaheuristics with the purely quantum approach. This comparison is expected to shed some light on the pressing question of whether quantum annealing is more efficient than classical methods, and, if so, to what degree.

Author Contributions: Conceptualization, Christos Papalitsas and Theodore Andronikos; Formal analysis,Christos Papalitsas and Theodore Andronikos; Methodology, Christos Papalitsas and Theodore

Andronikos; Supervision, Theodore Andronikos; Validation, Christos Papalitsas and Sofia Fanarioti; Writing - original draft, Konstantinos Giannakis and Georgia Theocharopoulou; Writing - review & editing, Konstantinos Giannakis, Georgia Theocharopoulou, Christos Papalitsas, Sofia Fanarioti and Theodore Andronikos.

Funding: This research is funded in the context of the project "Investigating alternative computational methods and their use in computational problems related to optimization and game theory," (MIS 5007500) under the call for proposals "Supporting researchers with an emphasis on young researchers" (EDULL34). The project is co-financed by Greece and the European Union (European Social Fund - ESF) by the Operational Programme Human Resources Development, Education and Lifelong Learning 2014-2020.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

QPU	Quantum Processing Unit
QUBO	Quadratic Unconstrained Binary Optimization
TSP	Traveling Salesman Problem
TSPTW	Traveling Salesman Problem with Time Windows

1. Feynman, R.P. Simulating physics with computers. *International Journal of Theoretical Physics Int J Theor Phys* **1982**, *21*, 467–488.
2. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review* **1999**, *41*, 303–332.
3. Grover, L. A fast quantum mechanical algorithm for database search. Proc. of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, 1996, 1996.
4. Nielsen, M.A.; Chuang, I.L. *Quantum computation and quantum information*; Cambridge university press, 2010.
5. Farhi, E.; Goldstone, J.; Gutmann, S.; Sipser, M. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106* **2000**.
6. Farhi, E.; Goldstone, J.; Gutmann, S.; Lapan, J.; Lundgren, A.; Preda, D. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science* **2001**, *292*, 472–475.
7. Choi, V. Minor-embedding in adiabatic quantum computation: I. The parameter setting problem. *Quantum Information Processing* **2008**, *7*, 193–209.
8. Messiah, A. Quantum mechanics, 1961.
9. Amin, M. Consistency of the adiabatic theorem. *Physical review letters* **2009**, *102*, 220401.
10. Kadowaki, T. Study of optimization problems by quantum annealing. *arXiv preprint quant-ph/0205020* **2002**.
11. Kadowaki, T.; Nishimori, H. Quantum annealing in the transverse Ising model. *Physical Review E* **1998**, *58*, 5355.
12. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *science* **1983**, *220*, 671–680.
13. Pakin, S. Performing fully parallel constraint logic programming on a quantum annealer. *Theory and Practice of Logic Programming* **2018**, *18*, 928–949.
14. Perdomo-Ortiz, A.; Dickson, N.; Drew-Brook, M.; Rose, G.; Aspuru-Guzik, A. Finding low-energy conformations of lattice protein models by quantum annealing. *Scientific reports* **2012**, *2*, 571.
15. Sarkar, A. Quantum Algorithms: for pattern-matching in genomic sequences **2018**.
16. Biamonte, J.; Wittek, P.; Pancotti, N.; Rebentrost, P.; Wiebe, N.; Lloyd, S. Quantum machine learning. *Nature* **2017**, *549*, 195.
17. Papalitsas, C.; Karakostas, P.; Andronikos, T.; Sioutas, S.; Giannakis, K. Combinatorial GVNS (General Variable Neighborhood Search) Optimization for Dynamic Garbage Collection. *Algorithms* **2018**, *11*, 38.
18. Rebentrost, P.; Schuld, M.; Wossnig, L.; Petruccione, F.; Lloyd, S. Quantum gradient descent and Newton's method for constrained polynomial optimization. *New Journal of Physics* **2019**.
19. D-Wave, S. Getting Started with the D-Wave System. Technical report, D-Wave Systems, 2019.

20. Boothby, K.; Bunyk, P.; Raymond, J.; Roy, A. Next-generation topology of D-Wave quantum processors. Technical report, D-Wave Systems, 2019.
21. Dattani, N.; Szalay, S.; Chancellor, N. Pegasus: The second connectivity graph for large-scale quantum annealing hardware. *arXiv preprint arXiv:1901.07636* **2019**.
22. Dattani, N.; Chancellor, N. Embedding quadratization gadgets on Chimera and Pegasus graphs. *arXiv preprint arXiv:1901.07676* **2019**.
23. Boros, E.; Crama, Y.; Hammer, P.L. Upper-bounds for quadratic 0–1 maximization. *Operations Research Letters* **1990**, *9*, 73–79.
24. Lewis, M.; Glover, F. Quadratic unconstrained binary optimization problem preprocessing: Theory and empirical analysis. *Networks* **2017**, *70*, 79–97.
25. Kochenberger, G.; Hao, J.K.; Glover, F.; Lewis, M.; Lü, Z.; Wang, H.; Wang, Y. The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization* **2014**, *28*, 58–81.
26. Lloyd, S.; Mohseni, M.; Rebentrost, P. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411* **2013**.
27. Hauke, P.; Katzgraber, H.G.; Lechner, W.; Nishimori, H.; Oliver, W.D. Perspectives of quantum annealing: Methods and implementations. *arXiv preprint arXiv:1903.06559* **2019**.
28. Glover, F.; Kochenberger, G. A Tutorial on Formulating QUBO Models. *arXiv preprint arXiv:1811.11538* **2018**.
29. Boros, E.; Hammer, P.L.; Tavares, G. Local search heuristics for quadratic unconstrained binary optimization (QUBO). *Journal of Heuristics* **2007**, *13*, 99–132.
30. Ushijima-Mwesigwa, H.; Negre, C.F.; Mniszewski, S.M. Graph partitioning using quantum annealing on the D-Wave system. Proceedings of the Second International Workshop on Post Moores Era Supercomputing. ACM, 2017, pp. 22–29.
31. Newell, G.F.; Montroll, E.W. On the theory of the Ising model of ferromagnetism. *Reviews of Modern Physics* **1953**, *25*, 353.
32. Lucas, A. Ising formulations of many NP problems. *Frontiers in Physics* **2014**, *2*, 5.
33. Neukart, F.; Compostella, G.; Seidel, C.; Von Dollen, D.; Yarkoni, S.; Parney, B. Traffic flow optimization using a quantum annealer. *Frontiers in ICT* **2017**, *4*, 29.
34. Martoňák, R.; Santoro, G.E.; Tosatti, E. Quantum annealing of the traveling-salesman problem. *Physical Review E* **2004**, *70*, 057701.
35. Warren, R.H. Adapting the traveling salesman problem to an adiabatic quantum computer. *Quantum information processing* **2013**, *12*, 1781–1785.
36. Warren, R.H. Small traveling salesman problems. *Journal of Advances in Applied Mathematics* **2017**, *2*.
37. Feld, S.; Roch, C.; Gabor, T.; Seidel, C.; Neukart, F.; Galter, I.; Mauerer, W.; Linnhoff-Popien, C. A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer. *arXiv preprint arXiv:1811.07403* **2018**.
38. Irie, H.; Wongpaisarnsin, G.; Terabe, M.; Miki, A.; Taguchi, S. Quantum annealing of vehicle routing problem with time, state and capacity. International Workshop on Quantum Technology and Optimization Problems. Springer, 2019, pp. 145–156.
39. Neven, H.; Denchev, V.S.; Rose, G.; Macready, W.G. Training a large scale classifier with the quantum adiabatic algorithm. *arXiv preprint arXiv:0912.0779* **2009**.
40. Garnerone, S.; Zanardi, P.; Lidar, D.A. Adiabatic quantum algorithm for search engine ranking. *Physical review letters* **2012**, *108*, 230506.
41. Cruz-Santos, W.; Venegas-Andraca, S.; Lanzagorta, M. A QUBO Formulation of the Stereo Matching Problem for D-Wave Quantum Annealers. *Entropy* **2018**, *20*, 786.
42. Babbush, R.; Love, P.J.; Aspuru-Guzik, A. Adiabatic quantum simulation of quantum chemistry. *Scientific reports* **2014**, *4*, 6603.
43. Crispin, A.; Syrichas, A. Quantum annealing algorithm for vehicle scheduling. 2013 IEEE International Conference on Systems, Man, and Cybernetics. IEEE, 2013, pp. 3523–3528.
44. Cai, B.B.; Zhang, X.H. Hybrid Quantum Genetic Algorithm and Its Application in VRP [J]. *Computer Simulation* **2010**, *7*.
45. Binder, K.; Young, A.; Fischer, K.; Hertz, J. Rev. Mod. Phys., 1986.
46. Young, A.P. *Spin glasses and random fields*; Vol. 12, World Scientific, 1998.

47. Nishimori, H. *Statistical physics of spin glasses and information processing: an introduction*; Number 111, Clarendon Press, 2001.
48. Venturelli, D.; Mandra, S.; Knysh, S.; O'Gorman, B.; Biswas, R.; Smelyanskiy, V. Quantum optimization of fully connected spin glasses. *Physical Review X* **2015**, *5*, 031040.
49. Titiloye, O.; Crispin, A. Quantum annealing of the graph coloring problem. *Discrete Optimization* **2011**, *8*, 376–384.
50. Venturelli, D.; Marchand, D.J.; Rojo, G. Quantum annealing implementation of job-shop scheduling. *arXiv preprint arXiv:1506.08479* **2015**.
51. Benedetti, M.; Realpe-Gómez, J.; Biswas, R.; Perdomo-Ortiz, A. Quantum-assisted learning of hardware-embedded probabilistic graphical models. *Physical Review X* **2017**, *7*, 041052.
52. Battaglia, D.A.; Santoro, G.E.; Tosatti, E. Optimization by quantum annealing: Lessons from hard satisfiability problems. *Physical Review E* **2005**, *71*, 066707.
53. Alom, M.Z.; Van Essen, B.; Moody, A.T.; Widemann, D.P.; Taha, T.M. Quadratic Unconstrained Binary Optimization (QUBO) on neuromorphic computing system. 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, 2017, pp. 3922–3929.
54. Pagano, G.; Bapat, A.; Becker, P.; Collins, K.; De, A.; Hess, P.; Kaplan, H.; Kyprianidis, A.; Tan, W.; Baldwin, C.; others. Quantum Approximate Optimization with a Trapped-Ion Quantum Simulator. *arXiv preprint arXiv:1906.02700* **2019**.
55. Farhi, E.; Harrow, A.W. Quantum supremacy through the quantum approximate optimization algorithm. *arXiv preprint arXiv:1602.07674* **2016**.
56. Venegas-Andraca, S.E.; Cruz-Santos, W.; McGeoch, C.; Lanzagorta, M. A cross-disciplinary introduction to quantum annealing-based algorithms. *Contemporary Physics* **2018**, *59*, 174–197.
57. Farhi, E.; Goldstone, J.; Gutmann, S. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028* **2014**.
58. Hadfield, S.; Wang, Z.; O'Gorman, B.; Rieffel, E.G.; Venturelli, D.; Biswas, R. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms* **2019**, *12*, 34.
59. Vyskocil, T.; Djidjev, H. Embedding Equality Constraints of Optimization Problems into a Quantum Annealer. *Algorithms* **2019**, *12*. doi:10.3390/a12040077.
60. Barán, B.; Villagra, M. A Quantum Adiabatic Algorithm for Multiobjective Combinatorial Optimization. *Axioms* **2019**, *8*, 32.
61. Oliveira, N.M.D.; Silva, R.M.D.A.; Oliveira, W.R.D. QUBO formulation for the contact map overlap problem. *International Journal of Quantum Information* **2018**, *16*, 1840007.
62. Rosenberg, G.; Vazifeh, M.; Woods, B.; Haber, E. Building an iterative heuristic solver for a quantum annealer. *Computational Optimization and Applications* **2016**, *65*, 845–869.
63. Langevin, A.; Desrochers, M.; Desrosiers, J.; Gélinas, S.; Soumis, F. A two-commodity flow formulation for the traveling salesman and the makespan problems with time windows. *Networks* **1993**, *23*, 631–640.
64. Dumas, Y.; Desrosiers, J.; Gelinas, E.; Solomon, M.M. An optimal algorithm for the traveling salesman problem with time windows. *Operations research* **1995**, *43*, 367–371.
65. Gendreau, M.; Hertz, A.; Laporte, G.; Stan, M. A generalized insertion heuristic for the traveling salesman problem with time windows. *Operations Research* **1998**, *46*, 330–335.
66. Da Silva, R.F.; Urrutia, S. A General VNS heuristic for the traveling salesman problem with time windows. *Discrete Optimization* **2010**, *7*, 203–211.
67. Sotiropoulos, D.; Stavropoulos, E.; Vrahatis, M. A new hybrid genetic algorithm for global optimization. *Nonlinear Analysis: Theory, Methods & Applications* **1997**, *30*, 4529–4538.
68. Pardalos, P.M.; Rodgers, G.P. Computational aspects of a branch and bound algorithm for quadratic zero-one programming. *Computing* **1990**, *45*, 131–144.
69. Lahoz-Beltra, R. Quantum genetic algorithms for computer scientists. *Computers* **2016**, *5*, 24.
70. Papalitsas, Ch.; Giannakis, K.; Andronikos, Th.; Theotokis, D.; Sifaleras, A. Initialization methods for the TSP with Time Windows using Variable Neighborhood Search. IEEE Proc. of the 6th International Conference on Information, Intelligence, Systems and Applications (IISA 2015), 6-8 July, Corfu, Greece, 2015.
71. Papalitsas, C.; Karakostas, P.; Kastampolidou, K. A Quantum Inspired GVNS: Some Preliminary Results. GeNeDis 2016; Vlamos, P., Ed.; Springer International Publishing: Cham, 2017; pp. 281–289.

72. Papalitsas, C.; Andronikos, T. Unconventional GVNS for Solving the Garbage Collection Problem with Time Windows. *Technologies* **2019**, *7*. doi:10.3390/technologies7030061.
73. Ohlmann, J.W.; Thomas, B.W. A compressed-annealing heuristic for the traveling salesman problem with time windows. *INFORMS Journal on Computing* **2007**, *19*, 80–90.
74. Vyskočil, T.; Pakin, S.; Djidjev, H.N. Embedding Inequality Constraints for Quantum Annealing Optimization. Quantum Technology and Optimization Problems; Feld, S.; Linnhoff-Popien, C., Eds.; Springer International Publishing: Cham, 2019; pp. 11–22.