Quantum and Digital Annealing for the Quadratic Assignment Problem

Philippe Codognet
JFLI
CNRS / Sorbonne University / University of Tokyo
Tokyo, Japan
codognet@is.s.u-tokyo.ac.jp

Daniel Diaz
CRI / University of Paris 1
Paris, France
daniel.diaz@univ-paris1.fr

Salvador Abreu
NOVA-LINCS / University of Évora
Evora, Portugal
spa@uevora.pt

***Abstract*—The Quadratic Assignment Problem is a a classical constrained optimization problem used to model many real-life applications. We present experiments in solving the Quadratic Assignment Problem by means of Quantum Annealing and Quantum-inspired Annealing. We describe how to model this classical combinatorial problem in terms of QUBO (Quadratic Unconstrained Binary Optimization) for implementing it on hardware solvers based on quantum or quantum-inspired annealing (D-Wave, Fujitsu Digital Annealing Unit and Fixstars Amplify Annealing Engine). We present performance result for these implementations and compare them with well established metaheuristic solvers on classical hardware, such as Robust Tabu Search and Extremal Optimization.**

I. Introduction

The Quadratic Assignment Problem (QAP) is a classical constrained optimization problem that has been studied for many years and considered as a classical benchmark for comparing various optimization methods. Introduced by [1] in the late 50's, QAP consists in considering n facilities and n locations, with some given *flows* of items between facilities and some given *distances* between locations. The problem is to minimize the overall cost consisting of the *flows* of items between facilities multiplied by the *distances* between locations. Many real-life applications can be model as QAP: scheduling, electronic board and microchip layout, communication network, chemistry, data center topology, see [2], [3] for more details. QAP is NP-hard and a large literature has been devoted to study this problem and develop efficient algorithms.

In this paper, we would like to use QAP to compare the performance of new specialized hardware based on "digital" or "quantum-inspired" annealing with respect to established metaheuristics running on classical hardware.

Following the theoretical work of [4] and [5], Quantum Annealing (QA) has been developed as a new method for solving combinatorial problems and hardware implementations on quantum machines have been developed since about a decade by D-Wave Systems [6], [7]. Less popular than the *gate-based model* in quantum computing, QA has the advantage of being less sensitive to noise and errors, and thus could be quite fit to the current NISQ (Noisy Intermediate Scale Quantum) era. Derived from simulated annealing [8], QA is based on the *quantum tunneling* effect to cross energy barriers and thereby avoids getting stuck in local extrema during the computation. Although coming from very different domains, QA is related to the Quadratic Unconstrained Binary Optimization (QUBO) formalism which has been used in combinatorial optimization for many years. Indeed, QUBO is the standard input language for quantum computers such as D-Wave or NTT's Coherent Ising Machine [9] and also for quantum-inspired dedicated hardware such as Fujitsu's Digital Annealer Unit (DAU) [10] and other systems developed by Hitachi, Toshiba, NEC and Fixstars Amplify (sometimes referred altogether as *digital annealing*).

Modeling optimization problems in QUBO and executing them on QA hardware is an emerging paradigm for combinatorial optimization [11], but could it be competitive with the best classical algorithms? The current literature only offers a partial answer to this issue as most papers about performance evaluation and comparison of QA solvers use combinatorial problems from graph theory (e.g., Max-Cut, Min-Cut, graph coloring, Traveling Salesman Problem,...) that are simple to model in QUBO and involve no or very few constraints. Modeling and benchmarking *constrained* optimization problems, where the objective function has to encode both some value to optimize and logical relations (constraints) between the problem variables is more complex. For instance, [12] compares a classical simulated annealing solver and three commercial quantum-inspired solvers (Fujitsu's Digital Annealing Unit, Toshiba's Simulated Bifurcation Machine and D-Wave's Hybrid solver - which combines quantum and classical computations) applied to three problems: Max-cut, Not-All-Equal 3-SAT and Sherrington-Kirkpatrick model. On small instances, D-Wave's Hybrid solver performs better, but on larger and more dense instances DAU performs better. Simulated annealing is outperformed by all other solvers. Small-size QAP instances are used in [13] to compare Fujitsu's DAU, a classical MIP solver (CPLEX) and a metaheuristic solver (EO-QAP [14]). Fujitsu's DAU beats CPLEX but not the local search metaheuristic: the DAU performs better than the *sequential* version of EO-QAP only in one instance out of six, and it always remains slower than the *parallel* version of EO-QAP on 32 cores [14].

The literature is thus lacking in assessing the effectiveness of quantum or "quantum-inspired" annealing as an alternative

way to solve complex constrained optimization problems as opposed to classical methods, in particular efficient metaheuristics. We would like in this paper to use the Quadratic Assignment Problem, in term of which several real-life applications can be stated, to further investigate the use of quantum annealing solvers for constrained optimization problems and compare their performance with metaheuristics-based solvers running on classical hardware.

This paper is organized as follows. Section II recalls the Quadratic Assignment Problem and two metaheuristics for solving it. Then Section III presents the QUBO formalism and its relation to the Ising model and quantum annealing. Section IV details how to encode global constraints such as `all-different` in QUBO and describes how to use such a constraint to model the Quadratic Assignment Problem in QUBO. We finally present, in Section V, some preliminary results obtained by implementing the QUBO models with quantum-inspired annealing and metaheuristics. A short conclusion ends the paper.

II. The Quadratic Assignment Problem (QAP)

Since its introduction in 1957, the Quadratic Assignment Problem (QAP) has been widely studied and several surveys are available [2], [3], [15], [16].

Take a set of *facilities* $\{1, \cdots, n\}$ and another of *locations* $\{1, \cdots, n\}$, both of size n. A flow of items between each pair of facilities is specified, as well as a distance between each pair of locations. The *Quadratic Assignment Problem* (QAP) consists in the assignment of each facility to a location so as to minimize the total sum of the flow-distance products.

A QAP instance of size n is defined by two $n \times n$ matrices $A = (a_{ij})$ and $B = (b_{ij})$. There are many formulations of QAP with different sets of variables and constraints, the most popular being as a permutation problem, as a 0-1 quadratic problem or as Mixed Integer Linear Programming (MILP) with different types of linearization, see [17]. In the *permutation formulation*, one has to find a permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ of $(1, 2, \ldots, n)$ that minimizes the objective function:

$$F(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \cdot b_{\pi_i \pi_j}$$

The A matrix represents flows between facilities and B represents distances between location. Therefore in facility location problems both A and B are usually symmetric but in different settings they could be asymmetric. Indeed, QAP can be used to model scheduling, chip placement and wiring on circuit boards, typewriter keyboard design, communication networks analysis, and many other applications [2], [18].

One defines the canonical form of a QAP instance as the size n, the $n \times n$ matrix A (flows) and the $n \times n$ matrix B (distances). Many QAP instances can be found in the QAPLIB library [19], both synthetic and stemming from real-world application scenarios.

Exact methods such as dynamic programming, cutting planes, or branch & bound, can solve medium-size instances, but those methods have problems with larger instances (e.g., $n > 20$). Incomplete methods such as *approximation algorithms* or *metaheuristics*, which can quickly produce good (but sub-optimal) solutions, can cope with large-size instances. Indeed, many metaheuristics have been successfully applied to the QAP: tabu search, simulated annealing, genetic algorithm, GRASP, ant colonies, etc, see for instance [3] for a general survey. The current trend is to specialize existing heuristics, to compose different meta-heuristics (hybrid procedures and portfolio approaches) and to use parallelism, but no single method outperforms all others on all benchmarks. In this paper, we will use two metaheuristics as a basis for comparison with quantum annealing: *Robust Tabu Search* (RoTS), which is well-known and widely used for QAP, together with *Extremal Optimization* (EO), which is quite simple, effective and very different from RoTS. The following sections briefly presents these methods, which are simple and versatile and perform well on a variety of QAP benchmarks.

A. Robust Tabu Search for QAP: RoTS

Robust Tabu Search has been proposed several decades ago in [20] as an extension of the Tabu Search metaheuristic [21]. Tabu Search is a neighborhood-based iterated local search method which selects, at each iteration, the best neighbor in order to explore the search space. So as to avoid being trapped in local minima, Tabu Search maintains an "adaptive memory" (the *Tabu List*), and forbids re-visiting a solution that is present in the Tabu List. It is interesting to note that Tabu Search can choose for the next iteration a "best neighbor" which in fact degrades the current value of the objective function, so long as it is not forbidden by the Tabu List. The Tabu List records solutions (or, more specifically, *attributes* of solutions) which are then banned for a set number of iterations (the *tabu tenure*). This is an important parameter that needs to be tuned for achieving the best performance: if the tabu tenure is too short, the method might not be able to avoid cycling and will re-visit previously *tabu* solutions, while a tabu tenure which is too long may interfere with the diversification of the search. An *aspiration mechanism* is thus often used in order to allow solutions that are forbidden by the Tabu List to be chosen if they pass a specific aspiration criterion, for instance if they provide an improvement in the objective function above some threshold. Robust Tabu Search specializes Tabu Search for the QAP and considers a transposition neighborhood, i.e., solutions are represented as permutations and the neighborhood of a permutation corresponds to all permutations where exactly two elements are swapped. It also provides an efficient procedure for examining the entire neighborhood and a dynamically changing tabu tenure for the Tabu List. Moreover, Robust Tabu Search relies on an interesting aspiration criterion: a move can be selected if the variables to be swapped have not occupied their target positions over the last t iterations, t being a tuning parameter.

B. Extremal Optimization for QAP: EO-QAP

The Extremal Optimization (EO) procedure was proposed by Boettcher and Percus [22] as a meta-heuristic to solve combinatorial optimization problems. It is based on the Bak-Sneppen model of *Self-Organized Criticality* (SOC) and inspired by the self-organization that can be found in many natural processes. Influenced by biological evolution, the key idea of EO is to consider *species* with a given *fitness* between 0 and 1 (0 being the worst fitness) and to design an *extremal* process that will progressively eliminates (or mutate) species with the least fitness in order to achieve a state where all species have a good fitness value (SOC). Therefore at each iteration, the species with the worst fitness is replaced, its value being randomly updated, and this modification will affect all other species connected to it as their fitness value will also be modified. Applying those general ideas to combinatorial optimization, EO is an iterative process that will check the current *configuration* (variable assignment), selects the variable with lowest fitness and modify its value with a random one. Obviously, selecting continuously the worst variable will amount to a deterministic behavior and may lead the algorithm to be trapped in a local minimum. This could be sidestepped by introducing some stochasticity as follows: variables are ranked in increasing order of fitness and a *Probability Distribution Function* (PDF) over rank k is used to choose the variable to be modified. In the original EO, a power-law PDF is used, as follows:

$$P(k) = k^{-\tau} \qquad (1 \leq k \leq n)$$

This PDF has a unique parameter τ which is problem-dependent. Indeed, a large variety of search strategies can be implemented with different values for τ: $\tau = 0$ will give a random walk, while $\tau \to \infty$ will give a greedy search. Giving a good value to τ will help EO to escape from local minima, as any variable is susceptible to mutate although the PDF will favor the worst ones. In [22], a default value for τ is proposed, depending on the number of problem variables n: $\tau = 1 + \frac{1}{ln(n)}$.

Over the years, various extensions have been proposed for the initial EO method, e.g., [23]–[25], and it has been used for solving large-scale optimization problems such as the TSP, graph bi-partitioning and coloring, and spin glasses problems.

EO has been recently adapted to solve QAP and found to perform very well, as reported in [14].

III. QUBO and Quantum Annealing

Although the roots of Quadratic Unconstrained Binary Optimization (QUBO) go back to pseudo-Boolean optimization in the late 60's, it was defined as a general modeling language for combinatorial problems only about 15 years ago [26]. The key interest of QUBO lies in the fact that it is very simple but also quite versatile and able to model a variety of combinatorial problems. Moreover, its similarity with the Ising model from statistical physics is an important aspect that will naturally connect with quantum annealing.

About a decade ago, [27] showed that many classical NP problems can be represented as Ising models and thus as QUBO. Indeed, many combinatorial optimization problems can be modeled as QUBO, e.g., graph-based problems (coloring, Max-Cut, Vertex Cover, Max-Clique, Maximum Independence Set), assignments or knapsack Problems, etc, see for instance [28], [29].

A. QUBO and the Ising Model

A QUBO problem consists of a vector of n Boolean variables $x_1, ..., x_n$ and a quadratic expression to minimize:

$$\sum_{i \leq j} q_{ij} x_i x_j$$

Thus a QUBO problem can be represented by a vector x of n Boolean variables and a $n \times n$ square matrix Q with coefficients q_{ij}. The matrix Q can be given in symmetric or upper triangular form, without loss of generality. Besides, a linear part can be included, as $x_i = x_i^2$ always holds for Boolean variables.

The QUBO formalism is very similar to the *Ising model*, that is used in statistical mechanics as a model of ferromagnetism. The Ising model is based on discrete variables that represent atomic spins that can have two possible states: +1 or -1. Interactions between neighboring spins is represented by a graph structure, e.g., a lattice. In the Ising model, the energy function, i.e., the Hamiltonian, is given by :

$$H = -\sum_i h_i \sigma_i - \sum_{i,j} J_{ij} \sigma_i \sigma_j$$

where the coefficients h_i represent the bias on the spin σ_i and the coefficients J_{ij} represent the strength of the coupling between spins σ_i and σ_j.

To transform an Ising model into a QUBO model, we just need to consider for each spin variables σ_i a Boolean variable x_i defined by $x_i = \frac{\sigma_i + 1}{2}$. Conversely, an optimization problem modeled as QUBO can be transformed into an Ising Hamiltonian, and the ground states of this Hamiltonian will therefore represent the minimal solutions of the QUBO problem. Those ground states can be computed by adiabatic quantum evolution in a physical device, an instance of which is quantum annealing [30].

B. Quantum Annealing

About two decades ago, [4] and [5] developed the theory of Quantum Annealing (QA) and D-Wave inc. started about one decade ago to develop computers based on QA with qubits implemented by superconducting Josephson junctions, see [31] for full details about those concepts in quantum statistical physics and a complete historical timeline. As described earlier, an optimization problem modeled as a QUBO can be transformed as an Hamiltonian in the Ising model. A QA computation starts by placing the initial system of qubits in a ground state (i.e., of minimal energy) that is easy to prepare and then progressively adding the *problem Hamiltonian* corresponding to the optimization problem while

gradually reducing the original Hamiltonian corresponding to the initial state. If the modification of the energy landscape is slow enough and there is no energy exchange with the outside world, the *Quantum Adiabatic Theorem* ensures that the system will remain in a ground state and thus that the final state has minimal energy. Therefore, the qubits in the final state represent a minimal solution of the optimization problem.

From an algorithmic point of view, QA can be seen as a variant of simulated annealing which differs by using *quantum tunneling* in order to escape local minima. Quantum tunneling is a phenomenon from quantum physics which is implemented in quantum annealing hardware and that makes it possible for the system to escape local minima by going through energy barriers as long as they are not too wide.

C. Quantum-inspired Annealing

Complementing the quantum computers developed by D-Wave, the idea to simulate quantum annealing on classical dedicated hardware has give rise in the last few years to the domain of *quantum-inspired* or *digital* annealing, see [11] for a complete list of such *Ising machines*. Several dedicated hardware systems exists, such as Fujitsu's Digital Annealer Unit (DAU) [10] and Hitachi's CMOS Annealing Machine [32]. Other systems include Fixstars Amplify Annealing Engine (AE) [33] based on a cluster of Graphics Processing Units (GPU), Toshiba's Simulated Bifurcation Machine (SBM) [34] and NEC Vector Annealer based on their own SX-Aurora TSUBASA Vector Engine [35].

All these systems use QUBO as their input language.

IV. The Quadratic Assignment Problem in QUBO

Although the formulation of the QAP in QUBO is well-known [13], [29], [36], we will present it in this section in a slightly different manner and with an emphasis on the notion of *constraints*, as exemplified in the Constraint Programming paradigm [37]. Constraint expressions can be introduced in QUBO models as quadratic *penalties* in the objective function, with an adequate encoding, and we will also give an intuitive meaning to the quadratic monomials that compose the penalties corresponding to each constraint.

A. Encoding Basic Constraints in QUBO

The penalty corresponding to a specific constraint can be defined by a quadratic expression which has its minimal value when the constraint is satisfied, e.g., an expression equal to zero upon constraint satisfaction and equal to a positive value otherwise. This penalty somehow represents the degree of violation of the constraint. Such a technique has indeed been in use for many years in constraint-based local search [38], [39].

Penalty expressions for a basic set of pseudo-Boolean constraints over a few Boolean variables can be found in [29]. For instance, the penalty for a constraint $x \leq y$ will be $x - xy$ and for $x + y = 1$ it will be $1 - x - y + 2xy$. Each penalty is added to the objective function with a specific coefficient. When many different constraints are combined into a single objective function, finding a good combination of penalty coefficients for combining the constraints can indeed be a problem it itself, see [40] for a concrete example and discussion on this topic.

Beyond simple pseudo-Boolean constraints, more complex constraints can also be encoded in QUBO models, e.g., the *One-Hot* constraint from the QA literature [41], which is equivalent to the `exactly-one` global constraint in the Constraint Programming community. This constraint enforces that, among n boolean variables $(x_1, \ldots, x_n)$, only one will take value 1 while all the others will be 0. It can be modeled by the pseudo-Boolean constraint $\sum_{i=1}^{n} x_i = 1$.

This constraint can be expressed in QUBO by the quadratic penalty $(\sum_{i=1}^{n} x_i - 1)^2$, which should be minimized down to zero in order for the constraint to be satisfied, and which can be simplified to:

$$-\sum_{i=1}^{n} x_i + 2\sum_{i<j} x_i x_j \tag{1}$$

The first term of this equation will try to maximize the number of x_i equal to 1, while the second term prevents any two x_i and x_j to be equal to 1 at the same time, thereby ensuring that a single variable x_i will be equal to 1.

B. The All-Different Constraint in QUBO

QAP is a permutation problem and each solution must represent a permutation over $\{1, \ldots, n\}$. This property can be expressed by the `all-different` constraint developed in Constraint Programming paradigm [42], [43].

Consider n variables $x_1, \ldots, x_n$ with values in $\{1, \ldots, n\}$, the constraint `all-different` $(x_1, \cdots, x_n)$ states that each variable has a value different from all others, thus $(x_1, \ldots, x_n)$ is a permutation of $(1, \ldots, n)$. To integrate this constraint in a QUBO model, we need to encode integer variables by Boolean variables and add corresponding constraints between these variables [44]. For each integer variable x_i, let us consider n Boolean variables x_{ij} such that $x_{ij} = 1$ if x_i has value j and $x_{ij} = 0$ otherwise. To enforce that each integer variable is represented correctly, i.e., that it has a single value, we need "One-Hot" constraints of the form $\sum_{j=1}^{n} x_{ij} = 1$. Moreover, in order to enforce the `all-different` constraint we also need to have "column constraints" of the form $\sum_{i=1}^{n} x_{ij} = 1$ Therefore the `all-different` over integers will be represented by $2 \times n$ pseudo-boolean constraints over Booleans:

$$\forall i \in \{1, n\} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall j \in \{1, n\} \quad \sum_{i=1}^{n} x_{ij} = 1$$

As detailed in [44], the quadratic penalties corresponding to all pseudo-Boolean constraints (cf. Equation 1) can be added together and simplified in order to give the overall penalty for the `all-different` constraint :

$$P_{\text{all-diff}} = -\sum_{i=1,j=1}^{n} x_{ij} + \sum_{k=1}^{n}\sum_{i<j} x_{ki}x_{kj} + \sum_{k=1}^{n}\sum_{i<j} x_{ik}x_{jk} \tag{2}$$

The first term will push to set to 1 as many variables x_{ij} as possible (n is the maximum), while the two other terms will forbid any original integer variable to have two values and any two different original integer variables to have the same value.

C. QAP in QUBO

For a QAP instance of size n consisting of two $n \times n$ matrices $F = (f_{ij})$ and $D = (d_{ij})$, we consider a QUBO model with n^2 Boolean variables x_{ik} such that $x_{ik} = 1$ if facility i is at location k and $x_{ik} = 0$ otherwise.

The objective function representing the flows between facilities is then expressed by:

$$H_0 = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ij}d_{kl}x_{ik}x_{jl}$$

As each facility has to be assigned to a different location, a penalty corresponding to an `all-different` constraint must be added (cf. Equation 2), with a penalty coefficient p. The final objective function to minimize is thus:

$$H_{QAP} = H_0 + pP_{\text{all-diff}}$$

V. Experimental Results

We first present in this section our experiments in executing QAP on the D-Wave quantum annealing machine and then we detail and compare the performance of QAP on digital annealing systems based on dedicated hardware (Fujitsu Digital Annealing Unit and Fixstars Amplify Annealing Engine) and metaheuristics on classical hardware (RoTS and EO-QAP), for both medium- and large-size instances.

A. D-Wave

Although each QUBO variable corresponds to a qubit in a QA system such as the D-Wave computers, it is difficult to implement models for which the maximal number of variables is close to the maximal number of qubits available, e.g., 5600 qubits for the D-Wave Advantage. Indeed, D-Wave systems do not implement a complete graph (fully connected) between qubits and therefore some qubits are also needed to encode extra "couplers" (connections between qubits). More precisely, in the D-Wave 2000X computer the *Chimera* connection graph allows for each qubit to be connected to 6 neighbors, while in the newer D-Wave Advantage the *Pegasus* connection graph allows for each qubit to be connected to 15 neighbors.

A transformation called *minor embedding* [45], [46] is thus needed in order to map a QUBO problem and the connection graph between variables (created by the different quadratic monomials in the objective function) on the physical hardware graph. The missing hardware connections can be simulated by using additional qubits, creating thus chains of *physical qubits* (bound to have identical values) for representing a single *logical qubit*. Thus, depending on the complexity of the QUBO model, the number of logical qubits (QUBO variables) available is different from the number of physical qubits. This depends in particular on the number of different quadratic monomials in the objective function, as each monomial will correspond to a connection between two different qubits. For instance on the D-Wave Advantage (5600 qubits), the maximal size of a complete (fully connected) graph between qubits, that can be embedded, is limited to 177 logical qubits [47].

For QAP, the QUBO models require n^2 variables for a problem of order n. In our experiments, we considered two small instances of order 12 that were also used in [13]: `rou12` and `had12`. These instances should be tractable in theory as they are modeled with 144 Boolean variables, thus below the limit of 177 logical qubits. Other instances would require more than 300 logical qubits. As the QUBO model uses an `all-different` constraint implemented as $2 \times n$ One-Hot constraints on n variables, it thus requires many point-to-point couplers in the QA hardware. Minor embedding of the QUBO model has to be performed for implementation on D-Wave hardware, and we used for this the heuristic tools from the D-Wave software suite. It has to be noted that those methods do not perform exhaustive search and thus can fail to find a minor embedding even if one theoretically exists. For QAP of order 12 with 144 boolean variables, the QUBO models with One-Hot constraints requires too many auxiliary qubits and the D-Wave software tools fail to provide a correct minor embedding. When run on the QPU, they produce infeasible solutions that do not satisfy the permutation constraint. Indeed, as minor embedding amounts to the creation of a chain with several physical qubits for encoding a single logical qubit (in order to provide more connections for this qubit), *chain breaks* can occur, leading to an infeasible solution. Only QUBO models of very small QAP instances can run on the QPU and provide feasible solutions which satisfy the permutation constraint.

A possible solution to this issue is to use post-processing methods in order to recover feasible solutions from infeasible ones. For instance the bit-flip heuristic algorithm [48] makes it possible to obtain feasible solutions up to size 20 but with a total run-time of several seconds, thus at the price of efficiency. Alternatively, the decomposition-based `Hybrid Solver` can be used on D-Wave systems. This solver mixes execution on classical hardware and on the QPU (for smaller subproblems) and can cope with problems up to 20,000 variables. D-Wave also proposes the `QBSolv` solver, a Tabu Search solver running entirely on classical hardware. It is, however, difficult for those solvers to reach the optimal solution and only `QBSolv` could do so within an extended running time, whereas the `Hybrid Solver` could not. For `rou12`, `QBSolv` will achieve a sub-optimal solution with an Average Percentage Deviation (APD) of 3% in 0.45 seconds and reach the optimum in 50 seconds. For `had12` it reaches a suboptimal solution with an APD of 2% in 1.25 seconds and the optimum in 84 seconds. With the `Hybrid Solver`, a suboptimal solution for `rou12` with an APD of 6% is reached in 3 seconds (of which 30 ms of QA on the QPU), and a suboptimal solution with an APD of 3.5% in 90 seconds, while for `had12` a suboptimal solution with an APD of 2%

is reached in 3 seconds and a suboptimal solution with an APD of 1% is reached in 90 seconds. As we shall see in the next section, those optimal solutions can be found in a few hundred of milliseconds with metaheuristic solvers or digital annealing.

B. Digital Annealing and Metaheuristics

In this section we present the performance results for the two classical metaheuristics methods presented in Section II and for the quantum-inspired digital annealers. As digital annealers implement (in hardware or software) the complete connection graph between all variables and can cope with 65,536 QUBO variables (Fixstars Amplify AE) or 100,000 QUBO variables (Fujitsu Digital Annealer version 3), they can take on larger instances of QAP than those experimented with on the D-Wave system. We implemented the QAP models on the digital annealers with the *Amplify SDK*, which is a middleware library for Ising machines. This python package provides basic functions to create QUBO matrices and to call several different solvers that are available as web services, in particular Fixstars Amplify AE, Fujitsu DA3 and D-Wave. In this way, we may use the same code (and thus create the same QUBO matrix) for both the Fixstars Amplify AE and Fujitsu DA3 and simply change the call to the adequate solver.

Table I details the performance results for metaheuristics and digital annealers on QAP instances. These where chosen to be of medium size (20 to 40 integer variables, therefore 400 to 1600 Boolean variables) and neither too easy nor too hard, i.e., taking a few seconds to solve in [14].
Entries for Extremal Optimization (EO-QAP) and Robust Tabu Search (RoTS) were taken on a Dell Precision 7820 with a Xeon Gold 6140 CPU at 2.3GHz using 64GB of RAM on a single core and correspond to the time needed to reach the Best Known Solution (BKS) of the problem instance[1]. Timing for DA3 is for the Fujitsu Digital Annealer Unit version 3 and AE for the Fixstars Amplify Annealer Engine (GPU-based), and considers only in the solving time (i.e., it does not include the web access time). Timings are in seconds and are the average of 50 runs for EO-QAP and RoTS and of 10 runs for DA3 and AE. When the solver cannot reach the optimal solution within 1 minute in *every run*, we indicate the percentage of times in which *it does*, within the time limit as well as the Average Percentage Deviation (APD), given by the average of the relative deviation *percentages*, which may be computed as $100 \times \frac{AvgSol-BKS}{BKS}$, where $AvgSol$ is the average over 10 obtained solutions.

Those results are obtained with the default parameters for RoTS and some parameter tuning for EO-QAP. For digital annealers, the Amplify SDK that we used does not provide any parameter tuning for the solver, although for the DA3 some tuning is possible when directly programming the hardware, e.g., for controlling the temperature cooling schedule or the parallel tempering. It could also be interesting to consider pausing in the annealing process [49]. This might possibly gain a factor of 2, but would most likely not qualitatively change the conclusions of this experiment.

For medium-size QAP it is clear that, at this stage, classical metaheuristics perform better than digital annealing on dedicated hardware. However, digital annealers achieve honorable results and the performance of this dedicated hardware approach is likely to significantly improve over the next few years.

C. Large-size QAP instances

Recently, the developers of Fujitsu's DAU reported in [50] experiments with large-size QAP instances ($n \geq 100$) and we can thus compare these results with the metaheuristics methods. Timings are in seconds and are the average of 50 runs for EO-QAP and RoTS and of 10 runs for DA3. On most large-size QAP instances, the solvers cannot find the optimal solution in a reasonable time (i.e., a few minutes), and therefore we report (as is done in [50]) the result obtained after a timeout of 300 seconds in the form of the APD (percentage over the optimal value) on the average of the runs. These performance results are shown in Table II, where entries for Extremal Optimization (EO-QAP) and Robust Tabu Search (RoTS) are taken on a Dell Precision 7820 with a Xeon Gold 6140 CPU at 2.3GHz using 64GB of RAM on a single core and entries for the DA3 are taken from [50]. As those large-size instances are hard to solve, it is necessary to tune the solver parameters in order to have good performance. This is done for RoTS and EO-QAP and we use, particularly for EO-QA, a restart strategy with short restarts on several instances. Unfortunately, [50] does not provide any information regarding parameter tuning for the DA3.

On these large-size instances the results are more mixed than for the mid-size instances. We can see that EO-QAP is much faster than other solvers on `esc128`, but also that DA3 is better on `sko100e` as it can find the optimal solution all the time whereas other solvers cannot. Finally, RoTS is better with the larger instances (`tho150`, `tai150b` and `tai256c`), as on those instances no solver can find the optimal solution before the timeout of 300 seconds, but RoTS comes closest to the optimum, then comes DA3 and lastly EO-QAP.

Although it is very interesting to observe that the performance of dedicated digital annealing hardware and of sequential execution of metaheuristics methods on a desktop computer are more or less equivalent, it should be noted that better performance can be achieved for QAP with metaheuristics methods on readily available parallel hardware, see for instance the results of [51] on a 64-core machine.

VI. Conclusion

We have detailed how to model the Quadratic Assignment Problem (QAP) in QUBO, so as to solve it with quantum annealing. Our aim was to compare – on medium-size and large-size QAP instances – the performance of quantum and quantum-inspired annealing hardware, relative to metaheuristic solvers running on classical hardware.

[1]Source code can be obtained at: https://github.com/didoudiaz/QAP-codes/

Problem	RoTS	EO-QAP	AE			DA3		
	time	time	success	time	APD	success	time	APD
had12	0.00	0.00	100%	0.30	-	100%	10.1	-
rou12	0.00	0.01	100%	0.15	-	100%	3.08	-
nug18	0.01	0.03	100%	2.24	-	0%	60.00	1.24%
esc32d	0.06	0.01	100%	1.17	-	0%	60.00	2.00%
rou20	0.11	0.69	30%	48.80	-	0%	60.00	0.93%
tai20a	0.21	1.03	0%	60.00	0.6%	0%	60.00	2.10%
chr22a	0.61	0.50	0%	60.00	4.2%	0%	60.00	3.48%
lipa40a	0.12	0.24	0%	60.00	2.3%			

TABLE I
METAHEURISTICS VERSUS QUANTUM-INSPIRED ANNEALERS
ON MEDIUM-SIZE QAP INSTANCES (TIMINGS IN SECONDS)

Problem	RoTS			EO-QAP			DA3 [50]		
	success	time	APD	success	time	APD	success	time	APD
sko100e	25%	268.9	0.003%	0%	300.0	0.081%	**100%**	108.6	-
esc128	100%	14.10	-	100%	**0.12**	-	100%	8.0	-
tho150	0%	300.0	**0.047%**	0%	300.0	0.334%	0%	300.0	0.145%
tai150b	0%	300.0	**0.221%**	0%	300.0	1.312%	0%	300.0	0.467%
tai256c	0%	300.0	**0.155%**	0%	300.0	0.236%	0%	300.0	0.212%

TABLE II
METAHEURSITICS VERSUS QUANTUM-INSPIRED ANNEALERS
ON LARGE-SIZE QAP INSTANCES (TIMINGS IN SECONDS)

These problems turn out to be too big for direct execution on quantum hardware (D-Wave) which cannot implement the complete connected graph between qubits. The problem constraints (permutation / `all-different`) end up requiring excessive additional qubits, even for a small QAP of order 12.

The current generation of quantum-inspired / digital annealers, such as Fujitsu DAU or Fixstars Amplify AE can cope and solve larger QAP instances. However, for medium-size instances, classical metaheuristic solvers such as Robust Tabu Search or Extremal Optimization perform better and are clearly faster in finding optimal solutions. For large-size QAP instances, the performance of Fujitsu's Digital Annealer (version 3) is comparable to that of the metaheuristic solvers, being globally not as good as Robust Tabu Search but better than Extremal Optimization.

Therefore, QUBO models and digital annealing *can* be used for modeling and solving complex constraint optimization problems and, as the hardware systems improve in the near future, it might become a very competitive technology, when compared to metaheuristics methods on classical hardware.

For quantum annealing, the fact that on D-Wave machines the execution time for an annealing sample is 130 microseconds (annealing + readout + cooling) in a system with 5000 qubits is encouraging, should the performance scale to larger systems. Indeed, if quantum annealers extend the number of available qubits by one or two orders of magnitude over the current systems, this computational approach will most likely challenge the best metaheuristic solvers on classical hardware.

REFERENCES

[1] T. C. Koopmans and M. Beckmann, "Assignment Problems and the Location of Economic Activities," *Econometrica*, vol. 25, no. 1, pp. 53–76, 1957.

[2] C. W. Commander, "A survey of the quadratic assignment problem, with applications," *Morehead Elect. Jnl. of Applicable Mathematics*, vol. 4, pp. MATH–2005–01, 2005.

[3] R. K. Bhati and A. Rasool, "Quadratic Assignment Problem and its Relevance to the Real World: A Survey," *Int. Jnl. of Computer Applications*, vol. 96, no. 9, pp. 42–47, 2014.

[4] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse ising model," *Phys. Rev. E*, vol. 58, pp. 5355–5363, Nov 1998.

[5] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, "A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem," *Science*, vol. 292, no. 5516, pp. 472–475, 2001.

[6] M. W. Johnson *et al.*, "Quantum annealing with manufactured spins," *Nature*, vol. 473, p. 194–198, 2011.

[7] P. I. Bunyk, E. M. Hoskinson, M. W. Johnson, E. Tolkacheva, F. Altomare, A. J. Berkley, R. Harris, J. P. Hilton, T. Lanting, A. J. Przybysz, and J. Whittaker, "Architectural considerations in the design of a superconducting quantum annealing processor," *IEEE Trans. on Applied Superconductivity*, vol. 24, no. 4, pp. 1–10, 2014.

[8] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[9] T. Inagaki and *et al.*, "A coherent ising machine for 2000-node optimization problems," *Science*, vol. 354, no. 6312, pp. 603–606, 2016.

[10] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. G. Katzgraber, "Physics-inspired optimization for quadratic unconstrained problems using a digital annealer," *Frontiers in Physics*, vol. 7, p. 48, 2019.

[11] N. Mohseni, P. McMahon, and T. Byrnes, "Ising machines as hardware solvers of combinatorial optimization problems," *Nature Reviews Physics*, 2022, published online 04/05/2022.

[12] H. Oshiyama and M. Ohzeki, "Benchmark of quantum-inspired heuristic solvers for quadratic unconstrained binary optimization," 2021.

[13] S. Matsubara *et al.*, "Digital annealer for high-speed solving of combinatorial optimization problems and its applications," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020, pp. 667–672.

[14] D. Munera, D. Diaz, and S. Abreu, "Solving the quadratic assignment problem with cooperative parallel extremal optimization," in *Evolutionary Computation in Combinatorial Optimization - EvoCOP 2016*, vol. 9595. Springer, 2016, pp. 251–266.

[15] R. E. Burkard, "Quadratic Assignment Problems," in *Handbook of Combinatorial Optimization (2nd edition)*, P. M. Pardalos, D.-Z. Du, and R. L. Graham, Eds. Springer New York, 2013, pp. 2741–2814.

[16] E. M. Loiola, N. M. M. de Abreu, P. O. B. Netto, P. Hahn, and T. M. Querido, "A survey for the quadratic assignment problem," *European Jnl. of Operational Research*, vol. 176, no. 2, pp. 657–690, 2007.

[17] L. Pitsoulis and P. M. Pardalos, "Quadratic Assignment Problem," in *Encyclopedia of Optimization*, C. A. Floudas and P. M. Pardalos, Eds. Springer, 2009, pp. 3119–3149.

[18] A. N. H. Zaied and L. A. E.-f. Shawky, "A Survey of Quadratic Assignment Problems," *Int. Journal of Computer Applications*, vol. 101, no. 6, pp. 28–36, 2014.

[19] R. Burkard, S. Karisch, and F. Rendl, "QAPLIB - a quadratic assignment problem library," *European Journal of Operational Research*, vol. 55, no. 1, pp. 115–119, 1991.

[20] É. D. Taillard, "Robust Taboo Search for the Quadratic Assignment Problem," *Parallel computing*, vol. 17, no. 4-5, pp. 443–455, 1991.

[21] F. W. Glover and M. Laguna, *Tabu Search*. Kluwer, 1997.

[22] S. Boettcher and A. G. Percus, "Nature's way of optimizing," *Artif. Intell.*, vol. 119, no. 1-2, pp. 275–286, 2000.

[23] ——, "Extremal Optimization: an Evolutionary Local-Search Algorithm," in *Computational Modeling and Problem Solving in the Networked World*. Springer US, 2003, vol. 21. [Online]. Available: http://arxiv.org/abs/cs.NE/0209030

[24] G.-Q. Zeng and Y.-Z. Lu, "Survey on computational complexity with phase transitions and extremal optimization," in *48th IEEE Conference on Decision and Control (CDC)*, 2009, pp. 4352–4359.

[25] M. Randall and A. Lewis, "Intensification Strategies for Extremal Optimisation," in *Simulated Evolution and Learning - 8th Int. Conf., (SEAL), Kanpur, India*. Springer, 2010, pp. 115–124.

[26] E. Boros, P. L. Hammer, and G. Tavares, "Local search heuristics for quadratic unconstrained binary optimization (QUBO)," *J. Heuristics*, vol. 13, no. 2, pp. 99–132, 2007.

[27] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, 2014.

[28] G. A. Kochenberger, J. Hao, F. W. Glover, M. W. Lewis, Z. Lu, H. Wang, and Y. Wang, "The unconstrained binary quadratic programming problem: a survey," *J. Comb. Optim.*, vol. 28, no. 1, pp. 58–81, 2014.

[29] F. W. Glover, G. A. Kochenberger, and Y. Du, "Quantum bridge analytics I: a tutorial on formulating and using QUBO models," *4OR*, vol. 17, no. 4, pp. 335–371, 2019.

[30] C. C. McGeoch, *Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice*. Morgan & Claypool, 2014.

[31] S. Tanaka, R. Tamura, and B. K. Chakrabarti, *Quantum Spin Glasses, Annealing and Computation*, 1st ed. USA: Cambridge University Press, 2017.

[37] F. Rossi, P. van Beek, and T. Walsh, Eds., *Handbook of Constraint Programming*, ser. Foundations of Artificial Intelligence. Elsevier, 2006, vol. 2.

[32] M. Yamaoka, T. Okuyama, M. Hayashi, C. Yoshimura, and T. Takemoto, "CMOS annealing machine: an in-memory computing accelerator to process combinatorial optimization problems," in *IEEE Custom Integrated Circuits Conference, Austin, TX, USA, 2019*. IEEE, 2019, pp. 1–8.

[33] Y. Matsuda, "Research and development of common software platform for ising machines," in *2020 IEICE General Conference*, 2020. [Online]. Available: https://amplify.fixstars.com/docs/_static/paper.pdf

[34] H. Goto, K. Tatsumura, and A. R. Dixon, "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear hamiltonian systems," *Science Advances*, vol. 5, no. 4, 2019.

[35] S. Dote, "NEC's initiative in quantum computing," 2021, presented at D-Wave Qubits conference 2021, october 2021.

[36] N. Nishimura, K. Tanahashi, K. Suganuma, M. J. Miyama, and M. Ohzeki, "Item listing optimization for e-commerce websites based on diversity," *Frontiers in Computer Science*, vol. 1, 2019.

[38] P. Codognet and D. Diaz, "Yet another local search method for constraint solving," in *Stochastic Algorithms: Foundations and Applications, International Symposium, SAGA 2001*. Springer, 2001, pp. 73–90.

[39] P. Galinier and J. Hao, "A general approach for constraint solving by local search," *J. Math. Model. Algorithms*, vol. 3, no. 1, pp. 73–88, 2004.

[40] T. Stollenwerk, E. Lobe, and M. Jung, "Flight gate assignment with a quantum annealer," in *Quantum Tech. and Optimization Problems*. Springer, 2019, pp. 99–110.

[41] S. Okada, M. Ohzeki, and S. Taguchi, "Efficient partition of integer optimization problems with one-hot encoding," Sept 2019.

[42] J. Régin, "A filtering algorithm for constraints of difference in csps," in *Proceedings of the 12th National Conference on Artificial Intelligence*. AAAI Press / The MIT Press, 1994, pp. 362–367.

[43] I. P. Gent, I. Miguel, and P. Nightingale, "Generalised arc consistency for the alldifferent constraint: An empirical survey," *Artif. Int.*, vol. 172, no. 18, pp. 1973–2000, 2008.

[44] P. Codognet, "Constraint solving by quantum annealing," in *ICPP Workshops 2021: 50th International Conference on Parallel Processing, August 2021*. ACM, 2021.

[45] V. Choi, "Minor-embedding in adiabatic quantum computation: I. the parameter setting problem," *Quantum Inf. Process.*, vol. 7, no. 5, pp. 193–209, 2008.

[46] ——, "Minor-embedding in adiabatic quantum computation: II. minor-universal graph design," *Quantum Inf. Process.*, vol. 10, no. 3, pp. 343–353, 2011.

[47] C. McGeoch and P. Farré, "The Advantage system: Performance update," Technical Report, D-Wave, 01-10-2021.

[48] M. Kuramata, R. Katsuki, and K. Nakata, "Larger sparse quadratic assignment problem optimization using quantum annealing and a bit-flip heuristic algorithm," 04 2021, pp. 556–565.

[49] M. Zielewski and H. Takizawa, "A method for reducing time-to-solution in quantum annealing through pausing," in *HPC Asia 2022*. ACM, 2022, pp. 137–145.

[50] H. Nakayama, J. Koyama, N. Yoneoka, and T. Miyazawa, "Description: Third generation digital annealer unit," Technical Report, Fujitsu,2021. [Online]. Available: https://www.fujitsu.com/global/documents/about/research/techintro/3rd-g-da_en.pdf

[51] M. Bagherbeik and A. Sheikholeslami, "Caching and vectorization schemes to accelerate local search algorithms for assignment problems," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, 2021, pp. 2549–2558.