Quantum approaches for WCET-related optimization problems

Gabriella Bettonte, Valentin Gilbert, Daniel Vert, Stéphane Louise, Renaud Sirdey

Quantum approaches for WCET-related optimization problems

Gabriella Bettonte,
Valentin Gilbert, Daniel Vert, Stéphane Louise, and Renaud Sirdey

Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France

gabriella.bettonte@cea.fr, valentin.gilbert@cea.fr,
daniel.vert2@cea.fr, stephane.louise@cea.fr, renaud.sirdey@cea.fr

Abstract. This paper explores the potential of quantum computing on a WCET[1]-related combinatorial optimization problem applied to a set of several polynomial special cases. We consider the maximization problem of determining the most expensive path in a control flow graph. In these graphs, vertices represent blocks of code whose execution times are fixed and known in advance. We port the considered optimization problem to the quantum framework by expressing it as a QUBO. We then experimentally compare the performances in solving the problem of classic Simulated Annealing (SA), Quantum Annealing (QA), and Quantum Approximate Optimization Algorithm (QAOA). Our experiments suggest that QA represents a fast equivalent of simulated annealing. Indeed, we measured the approximation ratio on the results of QA and SA, showing that their performances are comparable, at least on our set of simplified problems.

Keywords: WCETs · Quantum computing · QUBO · Combinatorial optimization · Quantum annealing · QAOA

1 Introduction

The interest given to quantum computing primarily comes from the ability of qubits to store a superposition state that reflects all the possible inputs and outputs of a given algorithm until a measurement is performed. This property is called *quantum parallelism* and can, in certain cases [21, 8], give a performance boost for quantum algorithms. However, the advantages of quantum computing do not come without caveats. Only some classes of problems can be solved by quantum computing, with a definite efficiency increase compared to classical computing [19]. One crucial research issue related to quantum computing is determining with precision which problems are better solved by means of quantum approaches [20].

This paper explores the potential of quantum computing by examining problems involved with determining the Worst-Case Execution Time (WCET) of a restricted set of programs. The problems arising in WCET evaluation cover a wide range of complexity classes, from undecidability in the general case to *NP*-hardness and polynomial-time solvability in some restricted cases [25]. As such, WCET evaluation appears to provide a relevant playground to put the quantum computing promise to the test. The execution

[1] Worst-Case Execution Time (of a program).

time of a program running on a machine depends on multiple factors, such as the initial system state, the hardware, and the input data. The validation of a real-time system requires knowing the WCET. However, the analysis of the exact WCET of a program is complicated by dynamic hardware mechanisms. A possible way to cope with the analysis complexity of WCET is by omitting such hardware mechanisms [15].

All the possible combinations of input data and execution paths need to be considered for computing the actual WCET of a program. This computation of an exact WCET is usually unfeasible with classic computers because the number of possible program paths can grow exponentially with the program size (notwithstanding decidability issues in the general case). Classical computing performs static analysis that approximates the WCET without actually executing the program. This approximation has to be an upper bound of the actual value of the worst-case execution time to assure the system's safeness. However, these WCET approximations should also not be overly pessimistic to avoid system over-dimensioning.

Yet, performing an exhaustive examination, even implicitly, of all possible program paths is a *sufficient* condition, even if not necessary, for understanding the worst-case scenario [15]. Thus, quantum computing could allow computing better worst-case-execution-time approximations, thanks to its promising computational power higher than classical computing. In a nutshell, the *program path analysis* method determines which sequence of instructions requires the highest execution time. The problem of determining a program's WCET generally is undecidable. Still, if there are no recursive function calls, dynamic structures, and unbounded loops in the program, it becomes decidable [11, 17]. Those strong hypothesis are enforced to be true in the field of real-time embedded systems, which is the primary target of WCET research.

In this paper, as a first step, we focus on a simplified program model: we consider only programs with IF-ELSE conditions (i.e., programs in which statically bounded loops have been unrolled), assuming uninterrupted executions and independence from the hardware. We assume the execution time of an instruction to be a constant: each instruction leads to a cache miss, and all data have to be fetched from the main memory. It is worth noticing that all these hypotheses make the problem solvable in polynomial time. Nevertheless, considering the limitation of existing quantum hardware, we argue that they represent an interesting model to be transposed to the quantum framework for benchmarking and that quantum computing approaches should be also evaluated against polynomial problems [24] rather than only on NP-hard ones [28]: Performing well on the former is presumably necessary to perform well on the latter. Furthermore, distance to optimality is of course easier to measure when attempting to solve polynomial-time problems on quantum hardware.

In particular, we examined well-known approaches for the estimation of worst-case execution times [14, 13, 25, 22]. Using Integer Linear Programming (ILP) problems, we can naturally describe the structure of our problem and the set of possible program paths, reducing the issue of estimating the WCET of a program into an optimization problem. The sum of the execution time of the executed *basic blocks* gives the cost function. Our goal is to find the maximum of this function. It is possible to adapt this optimization problem to the quantum computing framework through the penalty function method. The cost function and the constraints of the original problem are represented using linear and quadratic terms. Thus, the problem can be reformulated as a QUBO (Quadratic Unconstrained Binary Optimization) problem [16, 4] and solved by Quantum Approximate Optimization Algorithm (QAOA) and Quantum Annealing.

In this paper, we solve the problem with different methods: Quantum annealing on D-Wave machines, classical Simulated Annealing and Quantum Approximate Optimization Algorithm.The aim is to compare the performance of these methods (in term of optimization quality). Our tests suggest that D-Wave machines, in ideal cases, achieve the performances of classical Simulated Annealing while being much faster.

The paper is organized as follows. Section 2 provides the necessary preliminaries. Section 3 defines our problem and reformulates it in the form of a QUBO problem. Section 4 describes our evaluation parameters and the machines for the experiments. Section 5 collects the results of our experiments. Section 6 concludes the paper.

2 Combinatorial optimization

Combinatorial optimization computes the maximum or minimum of a function over a discrete domain. A combinatorial optimization problem is expressed as:

$$z^* = \underset{z \in S}{\operatorname{argmax}} \; f(z) \tag{1}$$

$$\begin{cases} g_l(z) = 0, & l = 1,...,L \\ h_m(z) < 0 & m = 1,...,M \end{cases} \tag{2}$$

where z is a discrete integer variables, $f(z)$ is the cost function, and $\mathcal{S}$ the set of decision variables satisfying the equality and inequality constraints given in (2) [26].

An integer linear programming (ILP) formulation is the mathematical formulation of an optimization problem in which variables are restricted to integer values and the constraints and cost function are linear [3]. ILP canonical form is expressed as:

$$\max \; c^T x \tag{3}$$

subject to

$$\begin{cases} Ax = b, \\ 0 \leq x, \qquad x \in \mathbb{Z}^n \end{cases} \tag{4}$$

A Quadratic Unconstrained Binary Optimization (QUBO) problem is a combinatorial problem defined through an upper triangular matrix $Q \in \mathbb{R}^{N \mathrm{x} N}$ and a vector x of binary variables. The goal of the optimization problem is to determine the vector of binary variables $\forall i, x_i \in \{0,1\}$ that minimizes (or maximizes) the objective function :

$$\sum_i Q_{ii} x_i + \sum_{i<j} Q_{ij} x_i x_j. \tag{5}$$

Using the penalty function method, we can rewrite any optimization problem into one without any constraints. For instance, given the equality constaint $g(z) = 0$, we can transform (1) into:

$$z^* = \underset{z}{\operatorname{argmax}} \; f(z) + \lambda g(z). \tag{6}$$

In this paper, we solve the QUBO problem to find the most expensive execution path by using quantum annealing (QA) and the Quantum Approximate Optimization Algorithm (QAOA). We compare the results with these of Classical Simulated Annealing (SA).

At this point, it is worth noting that any QUBO cost function can be transformed into a generalized 2D-Ising Hamiltonian with a simple transformation of variable $x_i = \frac{1+\sigma_i^z}{2}$ with $\sigma_i^z \in \{-1,1\}$:

$$\mathcal{H}_P = \sum_i^n h_i \sigma_i^z + \sum_{i<j}^n J_{ij} \sigma_i^z \sigma_j^z \tag{7}$$

2.1 Quantum annealing

Quantum annealing is a computational process that relies on the adiabatic theorem to solve combinatorial optimization problems. As a principle, it implements a time-dependent Hamiltonian composed of an initial Hamiltonian $\mathcal{H}_0$ whose ground state is easy to calculate and a final one tied to the cost function of the optimization problem as seen in equation (7): $\mathcal{H}_{\text{Ising}}(t) = A(t)\mathcal{H}_0 + B(t)\mathcal{H}_P$ such that $\mathcal{H}_{\text{Ising}}(0) = \mathcal{H}_0$ and $\mathcal{H}_{\text{Ising}}(\tau) = \mathcal{H}_P$ where τ is the optimal annealing time.

We choose $\mathcal{H}_0 = -\sum_i^n \sigma_i^x$ whose ground state corresponds to an equal superposition of the states of the computational basis. The adiabatic theorem states that if the time evolution is slow enough (i.e., τ is large enough), then the (global) optimal solution can be obtained with high probability. In practice, the final result is conditioned by the size of the spectral gap, the evolution time, the environmental or intrinsic decoherence effects, and the size of the coherence domain of the qubits on the chip.

If the quantum annealing can reach a minimum energy configuration, then the associated state vector solves the equivalent QUBO problem. Reformulating combinatorial problems in QUBO form preserves the underlying structure of the objective function [7].

2.2 Quantum Approximate Optimization Algorithm

The Quantum approximate optimization algorithm (QAOA)[5] is a quantum-classical algorithm used to solve optimization problems. It can be seen as an ansatz for the simulation of the Adiabatic process on a gate-based quantum computer. The approximation of the process is done using a parameter p stating the number of steps for the simulation: A p-depth QAOA consists of alternatively apply the two unitary propagators associated with both Hamiltonians $U(\mathcal{H}_P,\gamma)$ and $U(\mathcal{H}_M,\beta)$ on the initial state $|s\rangle$ which is a uniform superposition of all states of the computational basis.

$$|\psi\rangle = U(\mathcal{H}_M,\beta_p)U(\mathcal{H}_P,\gamma_p)...U(\mathcal{H}_M,\beta_1)U(\mathcal{H}_P,\gamma_1)|s\rangle \tag{8}$$

Each classical optimization round is used to optimize angles $\beta_1..\beta_p$ and $\gamma_1..\gamma_p$ to maximize the expectation value obtained from the run of the quantum circuit.

3 Problem designing: from control flow graphs to QUBO

We consider a simple micro-architecture [14] such that each basic block B_i of the program takes a constant time c_i to execute. A basic block is defined as a sequence of consecutive instructions. The flow of control enters into the block at the beginning and leaves at the end, without halt or possibility of branching except in the end. Let variable x_i be the execution count of the basic block B_i and N be the number of basic blocks in the program.

In this paper, we assume that $x_i \in \{0,1\}$, meaning that each basic block could be executed only once. The total execution time of the program is given by the linear expression:

$$\text{Program execution time} = \sum_{i}^{N} c_i x_i. \tag{9}$$

The problem has intrinsic constraints: not every execution flow is possible. For instance, if there is an IF-ELSE condition in our code, only the block respecting that condition is executed. Thus, one part of the program is ignored, depending on the input data. The WCET is given by the cost of the most expensive flow in terms of execution time. Notice that, in our model, we consider that the solution is unique. Our goal is to find the sequence of variables $x_0,\cdots,x_{N-1}$ such that it maximizes the cost function. To clarify all this, let us consider an example of a program with an IF condition: concretely, we have a sequential path that at some point splits into two branches and then reconnects again to the main path.

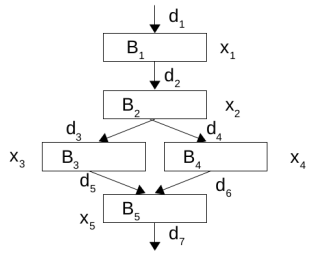


Fig. 1: Control flow diagram of a simple program structure

In this particular example we have only two possible paths of execution and the objective of our optimisation problem is to find the execution path that gives us the maximal cost:

$$\text{argmax} \;\; (c_1x_1+c_2x_2+c_3x_3+c_5x_5 \;\; , \;\; c_1x_1+c_2x_2+c_4x_4+c_5x_5) \tag{10}$$

We call x the vector representing the execution path, with $x_i = 1$ if the corresponding basic block B_i is executed and $x_i = 0$ otherwise. The solution is:

$$x = \begin{cases} \begin{bmatrix} 1 & 1 & 1 & 0 & 1 \end{bmatrix} \text{ if } (c_1x_1+c_2x_2+c_3x_3+c_5x_5 > c_1x_1+c_2x_2+c_4x_4+c_5x_5) \\ \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 1 \end{bmatrix} \text{ otherwise.} \end{cases}$$

In this example, the solution would be obtained simply considering the path that includes the most expensive branch of the IF condition. However, let us consider the constraints of this graph to compute the solution. The nodes of this graph are the blocks of code. The edges are the d_is, representing the action of *entering* or the *quitting* of a basic block. This graph represents the execution of a program, so we know for sure that the first and the last block have to be executed, so the edges entering and quitting

these nodes will be equal to one. Only one branch of the condition will be executed (if, for instance, $d_3=1$, then $d_4=0$). For this example, the set of constraints to consider is:

$$\begin{cases} d_1=1, \\ x_1=d_1=d_2 \\ x_2=d_2=d_3+d_4 \end{cases} \quad \begin{cases} x_3=d_3=d_5 \\ x_4=d_4=d_6 \\ x_5=d_5+d_6=d_7 \end{cases} \tag{11}$$

We transform this optimization problem to a QUBO moving the constraints to the cost function (penalty method). We omit the constant values in the cost function and we consider $x_i^2=x_i, \forall i$ because $x_i \in \{0,1\}$. Our optimization problem thus becomes:

$$\begin{aligned} &\text{argmax}_x(\textstyle\sum_i^5 c_i x_i - \lambda(1-x_3-x_4)^2) \\ &=\text{argmax}_x(\textstyle\sum_i^N c_i x_i - \lambda(1-x_3^2-x_4^2+2x_3x_4)) \\ &=\text{argmax}_x(c_1x_1^2+c_2x_2^2+(c_3+\lambda)x_3^2+(c_4+\lambda)x_4^2-2\lambda x_3x_4+c_5x_5^2)) \end{aligned} \tag{12}$$

Thus, for the considered problem, the matrix Q is:

$$Q=\begin{pmatrix} c_1 & 0 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 & 0 \\ 0 & 0 & c_3+\lambda & -2\lambda & 0 \\ 0 & 0 & 0 & c_4+\lambda & 0 \\ 0 & 0 & 0 & 0 & c_5 \end{pmatrix} \tag{13}$$

where $c_1,...,c_5 \in \mathbb{N}$ and $x=(x_1,x_2,x_3,x_4,x_5)^T$.

This paper focuses on a finite set of study cases to perform our experiments. The most basic program is not more than a linear sequence of instructions, thus with a deterministic and constant execution time, (Fig. 2(a)). We explored programs made by chains of consecutive IFs (Fig. 2(b)). This problem is interesting because, essentially, any program could be reduced to a loop with a condition in it. So, by unrolling loops, a program could be seen as a chain of conditions. Still, we want to underline that the problem is polynomial: the solution is easily founded by considering the most expensive branch at each IF.

Then we analyzed an expanded version by allowing the exclusive conditions to be more than two SWITCHes (Fig. 2(c)). Here again, the solution is given by the path that takes the most expensive branch at each IF condition, so the problem is still polynomially solvable. However, both examples give us the possibility of building an interesting benchmark for quantum computing.

Enlarging slightly more the focus on the targeted problem, we allowed the IFs blocks to have other nested IFs blocks inside them (Fig. 2(d)). This situation is slightly more complex than the previous ones because the paths need to be enumerated to find the actual worst-case path. All these cases of study are called *series–parallel graphs*.

The IFs chain and the SWITCH study case are easily generalizable from the matrix (13). The nested IF study case is more complex because it is not enough to choose the most expensive branch at each step. Thus, it is not possible to determine a pattern. We develop explicitly here the Q matrix only for a particular example. The Q matrix for the graph shown in Fig. 2(d) is:

$$Q=\begin{pmatrix} c_0 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_1 & -2\lambda & 2\lambda & 2\lambda & 0 \\ 0 & 0 & c_2+2\lambda & -2\lambda & -2\lambda & 0 \\ 0 & 0 & 0 & c_3 & -4\lambda & 0 \\ 0 & 0 & 0 & 0 & c_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_5 \end{pmatrix}$$

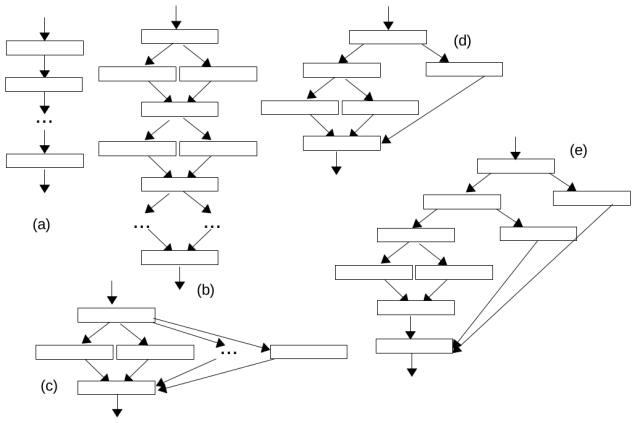


Fig. 2: Series–parallel graphs

3.1 Choice of the lambda value

Let $\{c_1,...c_k\} \in C$ be the set of weights for each node of the graph. $\{x_1,...x_k\}$ are boolean values defining if the basic block is executed. In the single IF case, the local cost function to the IF case is expressed as $(c_i+\lambda)x_i+(c_j+\lambda)x_j-2\lambda x_i x_j$ with c_i,c_j the weigths of each path and x_i,x_j the boolean values. The factor λ is appropriate if it follows the set of conditions:

$$\forall\ c_i,c_j \in \mathbb{R}^+ \text{we must have:}$$
$$c_i+\lambda > c_i+\lambda+c_j+\lambda-2\lambda \Leftrightarrow \lambda > c_j \tag{14}$$
$$c_j+\lambda > c_i+\lambda+c_j+\lambda-2\lambda \Leftrightarrow \lambda > c_i$$

For simplicity, we use a single λ for the whole expression, even if it contains multiple IF cases. We also consider that λ is an integer as each graph weight is integer. Therefore, λ should be greater than each weight of the graph yielding:

$$\lambda = \max(C)+1 \tag{15}$$

This choice of λ appears to behave well with the nested IFs and SWITCH too.

4 Benchmark Metrics and Computers

We compare the performances of SA, QA and QAOA while solving the optimisation problem of finding the most expensive execution path in the case tests we mentioned above: IF chains, SWITCH and nested IF. This section presents metrics, algorithms and quantum computers used for the benchmark.

4.1 Benchmark metrics

Problems presented in this paper are toy problems in which optimal solutions can be found in polynomial time. However, their simplicity allows us to fully evaluate and understand the results of our experiments. Before executing the experiments, each of

our optimization problems is transposed into their minimization form. To compare our results, we took into consideration two parameters:

- The approximation ratio (16). It represents the quality of the solution found compared to the whole energy landscape of the problem. E is the energy obtained from a single simulation, E_{min} is the energy of the optimal solution and E_{max} the energy of the worst solution.

$$r = \frac{E - E_{max}}{E_{min} - E_{max}} \tag{16}$$

  This parameter is interesting because, although the solution may not be the exact one, it could be very close.
- The optimal solution probability. It represents the proportion of optimal solutions (solutions having the energy E_{min}) obtained during the simulation.

4.2 Simulated Annealing

We configure the simulated annealing with an exponential decrease of temperature $T_1 = 0.95 * T_0$. In the following experiments, we set the thermal equilibrium at $T = 10^{-3}$. These iterations are fixed with the number of input variables n and may vary between $n^{0,5}$ to $n^{1,5}$. We consider SA running in a degraded mode where the number of iterations per temperature step is inferior to n. Each simulation is based on 100 runs of the SA to extract the mean of energy and the probability of getting the optimal solution.

4.3 Quantum annealing with D-Wave systems

Our experiments on D-Wave systems involve adiabatic quantum computing used to find the vector that minimizes the input cost function. At the moment, they represent the most advanced quantum machines having thousands of qubits. However, they require problems under the form of QUBOs and the topology of their chips limits their performance [23]. D-Wave systems used during our benchmarks are:

- **DW_2000Q_6** with 2048 qubits (2041 usable qubits) and 6 connections between each qubit (cf. Chimera topology).
- **Advantage_system4.1** with 5760 qubits (5627 usable qubits) and 15 connections between each qubit (cf. Pegasus topology).

For each experiment, results are computed with and without gauge inversion. The principle of a gauge inversion is to apply a Boolean inversion to the σ_i operators in our Hamiltonian. This technique preserves the optimal solution of the problem while limiting the effect of local biases of the qubits, as well as the machine accuracy errors [2]. Following the commonly used procedure (e.g. [1]), we randomly selected 10% of the physical qubits used as spin inversion for each instance. Each simulation is based on 1000 runs of D-Wave systems to extract the mean of energy and the optimal solution probability.

4.4 Simulation of QAOA

The simulation of QAOA is performed using the Qiskit library [6]. QAOA circuits are built from QUBO instances using penalty terms to express constraints. In our experiments, weights are specified with integers, hence $\gamma \in [0,2\pi]$ and $\beta \in [0,\pi]$. We did not find patterns to perform interpolation optimization as in [27]. We followed the

parameter fixing strategy [12] to set angles at p-depth. This method starts at $p=1$ and randomly generates 100 pairs of angles γ_1 and β_1. Then, we run a local optimizer on each of these pairs. We used COBYLA, a gradient-free optimizer. We run 1024 times the QAOA circuit at each optimization step to sample the mean expectation value corresponding to γ and β angles. At the end of the 100 optimization loops, we get 100 optimized pairs of angles. We select γ_1^* and β_1^* such as they minimize the value of the cost function. This process is then repeated at $p=n$, initializing the problem with $\gamma_1^*...\gamma_{n-1}^*$ and $\beta_1^*...\beta_{n-1}^*$ and 100 pairs of angles γ_n and β_n. For the simulation of the QAOA we used the *aer_simulator*, which provides a good speed performance.

5 Experimental results

To represent the cost of each basic block, we generated random integer, such that c_0,...,$c_n \in \{1,...,50\}$, and used them as input for our experiments. Each problem is designed to have only a single optimal solution, meaning that each branch candidate for solution should have a different global cost. In section 5.1 and section 5.2, each data point is smoothed over 30 randomly generated instances.

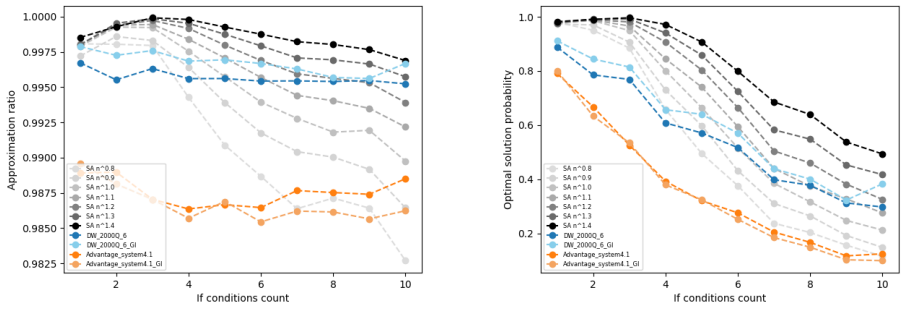
5.1 IF chains

Each IF chain (Fig. 2(a)) is composed of a succession of several IF conditions. This study case represents an ideal problem for the D-Wave as the corresponding QUBO matrix is sparse. Hence, the encoding on D-Wave systems does not require any duplication of qubits, neither for the 2000Q_6 nor the Advantage4.1 system. We start from one IF condition for this benchmark and grow the problem up to 10 IF conditions. A single block separates each IF condition block. We benchmark in Fig. 3 D-Wave systems against the resolution with classical simulated annealing. As the problem grows, QA seems to outperform SA progressively. For the IF chain, D-Wave system 2000_Q_6 systematically outperforms the Advantage_system4.1. Although the Advantage system is more recent than the 2000Q_6 system, it seems more sensitive to noise. We do not simulate QAOA for IF chains as it gives rather poor results compared to SA and D-Wave systems.

Fig. 4 shows the QAOA energy landscape at $p=1$ for a 3-if chain picked randomly. The heatmap exhibits many local minima, which impact the angle optimization of QAOA. This heatmap is more complex than heatmaps usually obtained for the well-studied Max-cut problem ([12],Fig. 5a). This complex energy landscape seems to be closely related to penalty terms that impact the whole energy landscape. Moreover, minimizing the mean energy does not always lead to an increased probability of getting the optimal solution.

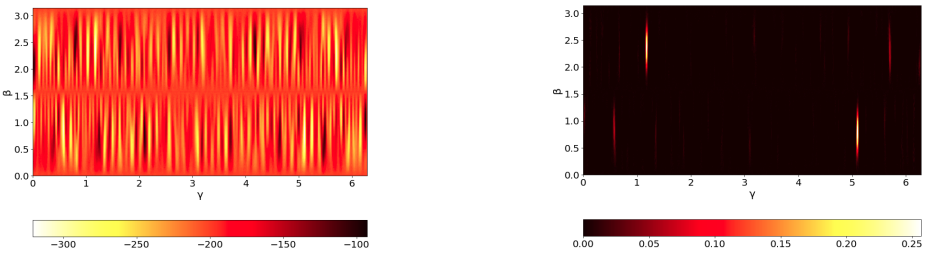
5.2 SWITCH

SWITCHes presented in Fig. 2(b) are harder to solve for D-Wave system since the plurality of choices leads to a dense matrix and requires higher connectivity between qubits. The impact of the density on QAOA is lower thanks to SWAP gates. In this execution case, we notice that the D-Wave Advantage4.1 performs better than the D-Wave 2000Q_6 for SWITCH cases 3 and 4. This improvement is due to the qubit duplication occurring on D-Wave 2000Q_6 whereas there is no qubit duplication on the Advantage4.1 for these instances (Fig. 5e). On larger instances (from 10 to 15 SWITCHes), the advantage of the



(a) (b)

Fig. 3: Benchmark of the QUBO resolution by D-Wave systems and SA for IF chains from 1 to 10 IF conditions. Each D-Wave simulation is done with and without GI (Gauge Inversion). n is the number of blocks in the if chain. SA number of iteration per temperature step is expressed according to n. a) Mean of approximation ratio. b) Mean of the probability to get the optimal solution.



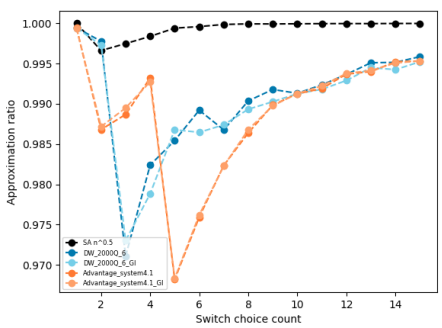
(a) Approximation ratio (b) Optimal solution probability

Fig. 4: QAOA heatmap at $p=1$ for a random 3 IFs chain. Approximation ratio is computed from the mean of the expectation value at angles γ and β.

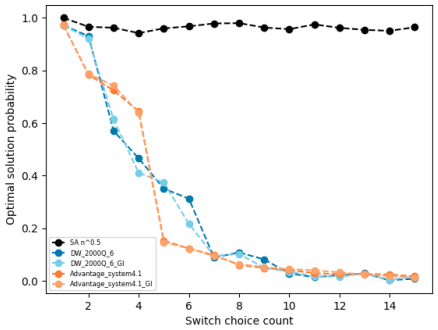
D-Wave Advantage4.1 provided by its number of connections is questionable. However, when the case is not ideal for D-Wave systems, the performances are poor against SA, even when SA is running under a degraded mode ($n^{0.5}$ iterations per temperature step).

5.3 Nested IFs

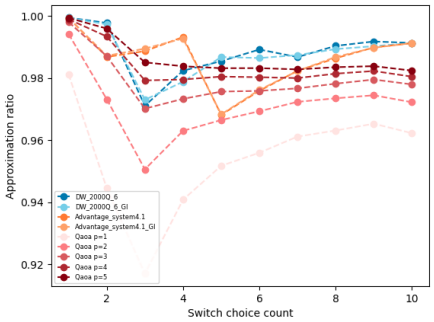
Nested IFs cases increase the density of the QUBO matrix. Table 1 shows two cases of nested IFs. The results are smoothed over 30 costs generated randomly. We compare the results obtained with SA, QA, and QAOA simulator. As for SWITCH cases, nested IF cases are difficult to be solved for D-Wave systems. They result to be not competitive against simulated annealing. QAOA simulations provide results of a lower performance compared to the D-Wave systems. The results obtained with QAOA start to decrease at $p=6$. This decrease may be due to the difficulty of COBYLA to optimize γ and β angles at this depth. As the optimization at depth p always starts from angles $\beta_1^*...\beta_{p-1}^*$



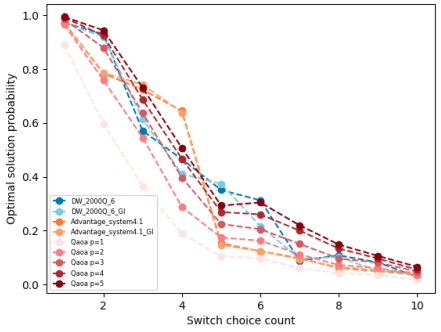
(a) Energy ratio SA and D-Wave



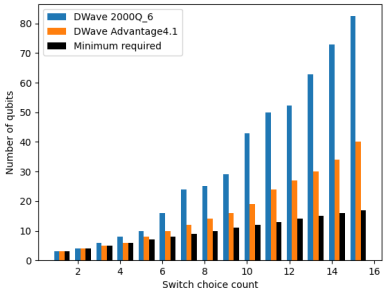
(b) Optimal solution probability



(c) Energy ratio QAOA and D-Wave



(d) Optimal solution probability



(e) Qubits used by D-Wave systems

Fig. 5: Comparison of switch instances solved with classical SA, D-Wave systems and QAOA simulator. (e) Mean of the number of qubits used by D-Wave systems to solve switch instances against the optimal number of qubits on ideal fully connected topology.

and $\gamma_1^*...\gamma_{p-1}^*$, the global optimization may be stuck at local minima imposed by the parameters found at $p-1$. The energy landscape of problems with penalty terms would deserves in-depth study to understand and find patterns of optimization of QAOA angles.

Table 1: Nested IFs simulation

Nested if results	**Case 1** (see fig.2d)		**Case 2** (see fig.2e)	
	Mean energy	**Opt sol prob**	**Mean energy**	**Opt sol prob**
SA $n^{0.5}$	0.985	0.66	0.999	0.934
SA $n^{0.6}$	0.986	0.69	0.999	0.923
SA $n^{0.7}$	0.989	0.76	0.999	0.956
D-Wave 2000Q	0.985	0.57	0.993	0.56
D-Wave Advantage	0.983	0.54	0.975	0.29
QAOA p=4	0.958	0.44	0.909	0.10
QAOA p=5	0.965	0.49	0.920	0.13
QAOA p=6	0.926	0.40	0.855	0.06

5.4 Real case

We provide a concrete application inspired by the bubble sort algorithm provided in the Mälardalen WCET research group [9][18]. The algorithm's goal is to sort an array of integers in ascending order. We consider an ideal scenario with an in-order, single-issue Arm processor similar to an M_0 with prefetched cache memory. Blocks composing the graph are built from the instructions obtained after the compilation of C code (using *gcc* Arm 8.3 with *-O2* level of optimization). The number of micro-instruction in each basic block defines its cost as given by our in-house WCET explorer with block identification and pipeline simulation. For the sake of simplicity, we limited to 3 the number of elements in the vector to sort. Table 2 shows the obtained result while solving the problem with SA and QA. From Table 2 we conclude that the 2000Q system performs quite well, even duplicating 4 qubits. The Advantage system, while duplicating only 1 qubits performs poorly. We may conclude that duplicating on Advantages machines downgrade drastically the performances of the machine.

Table 2: Real case simulation

Method	**Mean energy**	**Opt sol prob**	**qubits used**	**Max qubits duplication**
SA $n^{0.5}$	0.944	0.58	/	/
SA n^{1}	0.972	0.79	/	/
SA $n^{1.1}$	0.990	0.91	/	/
D-Wave 2000Q_6	0.987	0.868	15	1
D-Wave Advantage4.1	0.901	0.325	12	1

6 Conclusion

In this paper, we explored the potential of quantum computing in dealing with the combinatorial optimization problem of finding the most expensive execution path in a graph on a restricted set of simple instances. Our work offers an example of using quantum computing in a new application field. The performances we obtained using

D-Wave machines and the QAOA do not suggest that quantum computing is the silver bullet solution that will replace the classical computing solid and road-tested techniques. Still, we claim the results we obtained are encouraging enough to explore further the potential of quantum computing on the proposed problem (for instance, the backtracking quantum algorithm). Indeed, our experiments suggest that quantum computing gives us a fast equivalent of simulated annealing for ideal problems, such as the IFs chains.

We can draw some conclusions from the experience of this paper on D-Wave machines. Their topology is quite limiting, and it is not straightforward to adapt the considered optimization problem to it. We noticed that the Advantage4.1 machine outperforms the 2000Q machine when the considered problems involve qubit duplications on 2000Q device and not on the Advantage4.1 machine. It may be worth exploring the performances of these machines while solving problems that need more qubit duplications to be adapted to both topologies. The goal is to find parameters that show a priori which D-Wave machine is the most suitable to solve the considered problem. The chain of IFs is ideal for the topology of D-Wave systems, and their results are competitive with SA. Concretely, SA on 5600 variables at n^2 would need billions of evaluations of the cost function against millisecond runtime for D-Wave systems. However, we stress again that the problem is polynomially solvable using methods other than SA, and the interest is the possibility of building a benchmark for quantum computers.

We can as well draw some conclusions about QAOA. At $p=5$ the performances of QAOA are almost the same as D-Wave. QAOA does not have the problem of qubit duplication since we can circumvent the chip's topology with SWAP gates. Still, we performed our experiments on a simulator and not on a physical machine as D-Wave systems. An idea to investigate is to restrict the search on the feasible subspace of the problem [10]. In our case, this would be a subspace where every solution preserves the flow constraint.

Another aspect that may be explored in the future is the behavior of our models considering the effect of cache memories. Additionally, our choice of λ has no guarantees of being optimal, even if it provides good results. As a perspective, it may be interesting to perform a pre-processing to find, through simulations, the optimal λ. However, the effort of finding the optimal λ could overcome its benefits.

Acknowledgements

The authors acknowledge financial support from the AIDAS project of the Forschungszentrum Julich and CEA.

References

1. T. Albash and D. Lidar. Demonstration of a scaling advantage for a quantum annealer over simulated annealing. *Physical Review X*, 8, 2018.
2. S. Boixo, T. Albash, F. M. Spedalieri, N. Chancellor, and D. A. Lidar. Experimental signature of programmable quantum annealing. *Nature communications*, 4:2067, 2013.
3. Bradley, Hax, and Magnanti. Applied mathematical programming.
4. C. C. Chang, C. C. Chen, C. Koerber, T. S. Humble, and J. Ostrowski. Integer programming from quantum annealing and open quantum systems, 2020.
5. E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.
6. J. Gambetta and et al. Qiskit: An open-source framework for quantum computing, 2022.

7. F. Glover, G. Kochenberger, and Y. Du. A tutorial on formulating and using qubo models. *arXiv preprint arXiv:1811.11538*, 2018.
8. L. K. Grover. A fast quantum mechanical algorithm for database search, 1996.
9. J. Gustafsson, A. Betts, A. Ermeddahl, and B. Lisper. The mälardalen wcet benchmarks - past, present and future. In *Proceedings of the 10th International Worshop on Worst-Case Execution Time Analysis*, July 2010.
10. S. Hadfield, Z. Wang, B. O'Gorman, E. Rieffel, D. Venturelli, and R. Biswas. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms*, 12(2):34, Feb. 2019.
11. E. Kligerman and A. D. Stoyenko. Real-time euclid: A language for reliable real-time systems. *IEEE Transactions on Software Engineering*, SE-12(9):941–949, 1986.
12. X. Lee, Y. Saito, D. Cai, and N. Asai. Parameters fixing strategy for quantum approximate optimization algorithm. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, Oct. 2021.
13. Y.-T. Li, S. Malik, and A. Wolfe. Efficient microarchitecture modeling and path analysis for real-time software. In *Proceedings 16th IEEE Real-Time Systems Symposium*, pages 298–307, 1995.
14. Y.-T. Li, S. Malik, and A. Wolfe. Cache modeling for real-time software: beyond direct mapped instruction caches. *2011 IEEE 32nd Real-Time Systems Symposium*, 0:254, 01 1996.
15. J.-C. Liu and H.-J. Lee. Deterministic upperbounds of the worst-case execution times of cached programs. In *1994 Proceedings Real-Time Systems Symposium*, pages 182–191, 1994.
16. A. Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2:5, 2014.
17. P. Puschner and C. Koza. Calculating the maximum execution time of real-time programs. *Real-Time Systems*, 1:159–176, 01 1989.
18. M. W. research group. Wcet benchmarks, `http://www.mrtc.mdh.se/projects/wcet/benchmarks.html`.
19. T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer. Defining and detecting quantum speedup. *Science*, 345(6195):420–424, Jul 2014.
20. T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer. Defining and detecting quantum speedup. *Science*, 345:420–424, 2014.
21. P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, Oct 1997.
22. H. Theiling and C. Ferdinand. Combining abstract interpretation and ilp for microarchitecture modelling and program path analysis. In *Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No.98CB36279)*, pages 144–153, 1998.
23. D. Vert, R. Sirdey, and S. Louise. On the limitations of the chimera graph topology in using analog quantum computers. In *Proceedings of the 16th ACM International Conference on Computing Frontiers*, pages 226–229. ACM, 2019.
24. D. Vert, R. Sirdey, and S. Louise. Benchmarking quantum annealing against "hard" instances of the bipartite matching problem. *SN Comput. Sci.*, 2:106, 2021.
25. R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. F. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström. The worst-case execution-time problem—overview of methods and survey of tools. *ACM Trans. Embed. Comput. Syst.*, 2008.
26. M. Zaman, K. Tanahashi, and S. Tanaka. Pyqubo: Python library for mapping combinatorial optimization problems to QUBO form. *CoRR*, abs/2103.01708, 2021.
27. L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X*, 10(2), June 2020.
28. Özlem Salehi, A. Glos, and J. A. Miszczak. Unconstrained binary models of the travelling salesman problem variants for quantum optimization. *Quantum Information Processing*, 21(2), jan 2022.