IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 11 August 2023, accepted 27 August 2023, date of publication 7 September 2023, date of current version 14 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3312990

RESEARCH ARTICLE

Assessing the Effectiveness of Non-Turing Computing Paradigms

LORENZO ROCUTTO[1,2,*], (Member, IEEE), MARCO MARONESE[1,2,*], (Member, IEEE), FABIO LORENZO TRAVERSA[3], SERGIO DECHERCHI[4], AND ANDREA CAVALLI[1,2]

[1]Computational and Chemical Biology, Fondazione Istituto Italiano di Tecnologia, 16163 Genoa, Italy
[2]Department of Pharmacy and Biotechnology, University of Bologna, 40126 Bologna, Italy
[3]Memcomputing Inc., San Diego, CA 92121, USA
[4]Data Science and Computation, Fondazione Istituto Italiano di Tecnologia, 16163 Genoa, Italy

Corresponding authors: Sergio Decherchi (sergio.decherchi@iit.it) and Fabio Lorenzo Traversa (ftraversa@memcpu.com)

*Lorenzo Rocutto and Marco Maronese contributed equally to this work.

ABSTRACT In recent years the technological limits inherently present in the classical Turing paradigm of computation have sparked the development of innovative solutions based on quantum devices or analog-digital mixed approaches often based on the time evolution of differential equations. Such promising machinery require accurate analysis to understand if and how they will be able to perform better than classical approaches in solving hard optimization problems. Here we challenge two machines representative of the quantum annealing and differential equations approaches, namely D-Wave and Memcomputing by devising a benchmark of three well known hard optimization problems from the realms of number theory, optimal transport and optimal scheduling. We introduce the Mean First Solution Time, a novel metric for accurately comparing performances, and take as baseline the classical Gurobi solver. We show that performances of both solvers are heavily dependent on the selected set of internal parameters. Results shed lights on the advantages and current limits of each paradigm and give a perspective on possible future developments.

INDEX TERMS Factorization problem, in-memory computation, non-linear dynamical systems, non-turing computation, NP optimization problems, optimal transport, quantum annealing.

I. INTRODUCTION

The Von Neumann architecture [1], and hence the Turing paradigm [2], has dominated computing since the time of Charles Babbage [3]. Moore's law [4] predicts an exponentially increasing number of transistors which can be translated to a sustained growth of the computing power. In recent years, this increase has come from specialized and highly parallel accelerators, chiefly graphics processing units (GPUs). The quest for ever-increasing performance now requires new approaches with more efficient energetic profiles and a reduced environmental impact. Alternative architectures need to be investigated in depth towards a more concrete green-computing implementation. Notwithstanding the traditional Turing machines may still be faster, technological signs of progress may quickly change this situation, providing the community with faster architectures that are energetically more sustainable.

Non traditional hardware/software platforms are today available, an example being quantum-gate-based computing assets [5]. These machines introduced by Deutsch [6], are the quantum generalizations of the class of Turing machines. As emerged from the study by Bernstein and Vazirani [7], which introduced the concept of quantum complexity classes, quantum computers can solve certain problems exponentially faster than classical computers. Quantum-gate-based computers are indeed universal computers; Adriano Barenco et al. [8] describe the quantum counterpart of the classical logical gates, called quantum gates, and provide a universal quantum gate set. These quantum gates are implemented through photonic qumodes [9], trapped ions qubits [10], [11], and superconducting qubits. Currently many important players as Google [12] and IBM [13] among others [14] are dedicating significant resources to the implementation of such computing systems. It is debated when the *quantum gold*

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Huang.

rush [15] will generate practical utility [16], [17], [18], [19]. In [16] the Google team claimed to have reached Quantum Supremacy with their Sycamore QPU, while in [17] the IBM team contended such claim. Later, [18] and [19] showed how a modern supercomputer can match the aforementioned ''quantum supremacy'' performances.

Besides this quantum, Turing-gate-based machinery, there are other non-classic alternatives such as the Quantum Annealer (D-Wave Systems [20]). There are also different ways to use classical phenomena to perform calculations, e.g. the Memcomputing Machine (Memcomputing Inc. [21]), the Simulated Bifurcation Machine (Toshiba [22]), and approaches such as the Ising machine [23] and p-bits [24]. To fairly evaluate these different computing paradigms, proper benchmarks must be defined to assess the potential of these architectures to solve difficult tasks faster than conventional computers. For instance in [25] benchmarks are defined for assessing the speed-up against classical solvers; in [26] and [27] methods are described to evaluate quantum computing performances.

Here, we critically assess two commercially widely available and paradigmatic non-Turing approaches and compare them to a classical solver (see Figure 1). D-Wave and Memcomputing can solve equivalent problems and they represent a quantum (not gate-based) and classical (not quantum) approach, respectively. D-Wave solves Quadratic Unconstrained Binary Optimization problems (QUBO) and is an adiabatic quantum annealer (AQC). Memcomputing solves Integer Linear Programming problems (ILP) by mapping them to a physical circuit, with the physical circuit evolution emulated via a software. D-Wave is the most mature representative of the quantum annealing strategy. Memcomputing represents an approach based on differential equations with digital read-out (similar in spirit to the bifurcation machine [22]). The two approaches can be directly compared with an established baseline, namely the Gurobi suite of optimization methods [28].

We benchmark these approaches on three difficult, well-known, and combinatorial problems of broad interest that can be expressed in ILP (Memcomputing) and QUBO (D-Wave) format: the Semiprime Factorization problem (FP), the Hard-Assignment Gromov-Wasserstein problem (GWP), and the Capacitated Helicopter Routing Problem (CHRP). The security of large part of the public key cryptography (RSA [29]) is based upon the assumed intractability of FP. GWP is a particularly hard example of the optimal transport theory [30], [31] and is an instance of the well-known Quadratic Assignment Problem [32], a fundamental combinatorial problem. CHRP is an industrial optimization problem concerning the route scheduling of helicopters.

For an accurate comparison, we introduce the concept of Mean First Solution Time (MFST) which is the expected waiting time to obtain a first solution of the problem; we also give a finite sample estimator of this quantity. It is thus directly proportional to the expected total amount of monetary budget required to solve the problem. We compare this novel metric to the widely used probability-normalized Time To Solution metric (TTS). TTS has been used to evaluate performances of quantum annealers in [33], [34], and [35]. We also show that the performances of the solvers strongly depend on the selected set of internal parameters. This is the first time AQCs and Memcomputing machines are compared to each other on such a comprehensive set of industrially and mathematically relevant problems, and it is the first time that the dependence of their performances on internal parameters is analyzed in detail.

In the following Methods section, we introduce in more detail the computing machinery and the three benchmark problems. In the Results section, the obtained outcomes are presented and discussed in terms of scaling, parameters dependency and latency. In the Discussion and Conclusions section we summarize the main results and provide a final discussion.

II. METHODS

Here we briefly describe the Non-Turing machines used, the problems, how problems were implemented, and the formal definition of the Mean First Solution Time. Further details on these aspects are available in Supplementary Material (SM).

A. COMPUTING MACHINERY

1) MEMCOMPUTING

Memcomputing is the name given to an emerging computational paradigm that exploits the evolution of a circuit based on memristors to perform computations. It should not be confused with the in-memory or near-memory computing paradigm which was conceived to avoid most of the costs of moving data by processing directly within the memory subsystem [36]. In fact, Memcomputing is a non-Turing paradigm that does not exploit the Von Neumann architecture. It does not have a dedicated memory component but rather exploits the evolution of a physical system. The dynamical evolution of this system, therefore, plays the role of a fictitious memory.

Memcomputing devices perform computations harnessing the nonlinear dynamics of a physical system. Such concept was pioneered by Chua and Lin [37]. The Chua's approach allows to solve nonlinear optimization problems [37] through possibly such nonlinear dynamics. One of the key differences between the Chua's approach and Memcomputing is that while the former is a fully analogical method, the latter exploits logical gates, hence the dynamical evolution is used to support a fully bit-based solver.

Another computational device based on the dynamic of nonlinear systems has been recently devised. This is the Simulated Bifurcation Machine (SBM) developed by Toshiba [22] which is inspired by quantum principles. The SBM simulates a classical system of nonlinear oscillators that produces bifurcations in the phase space, enabling the simulation of a spin-glass system. As SBM is a fully digital solution this system is currently the most closely related to Memcomputing.

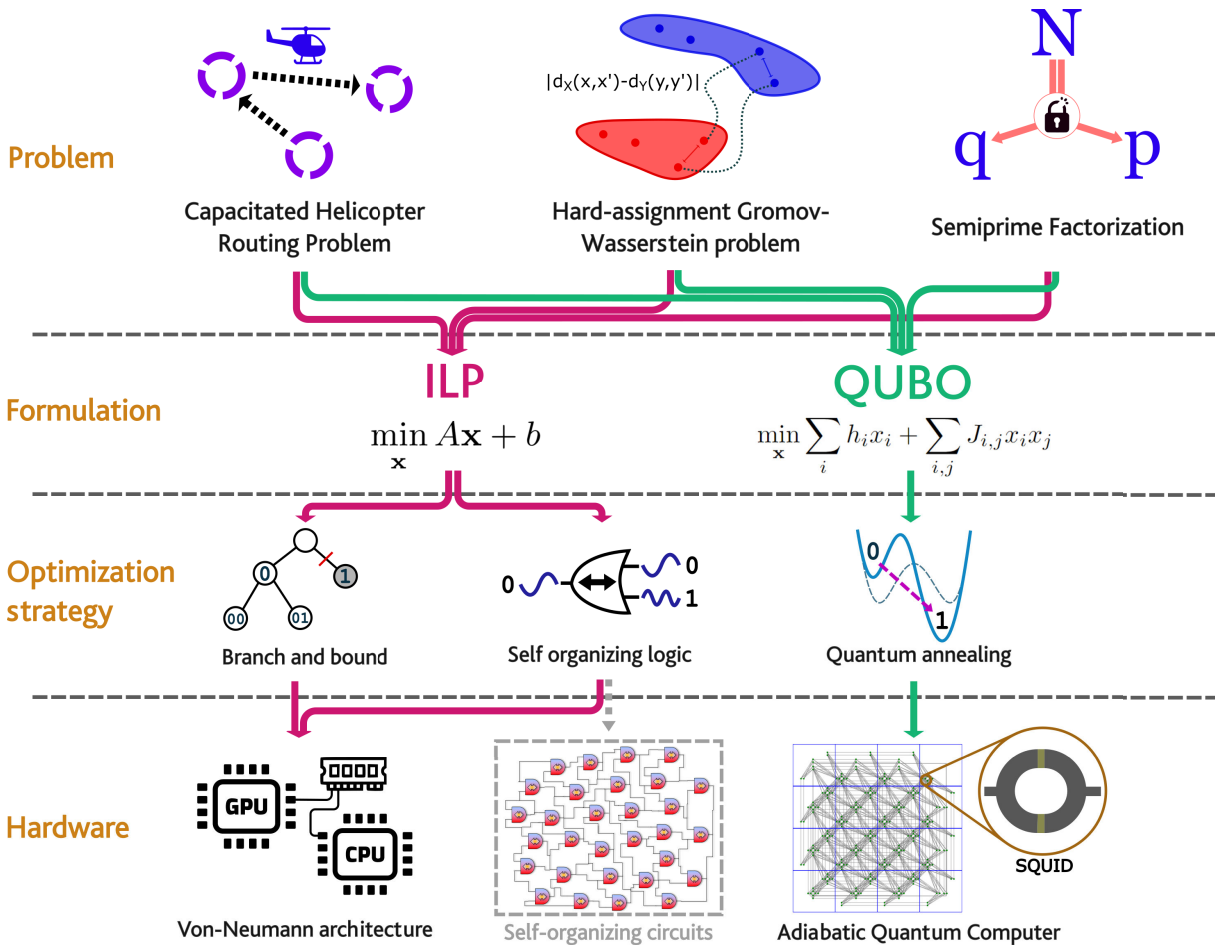

FIGURE 1. From the problem to the computing hardware. Three problems are formulated in ILP and QUBO forms and solved with three different solvers: i) the Gurobi optimization software based on branch and bound and other heuristics; ii) a Virtual Memcomputing Machine exploiting self-organizing logic; and iii) a Quantum Annealer. These solvers are physically implemented on hardware based on the Von-Neumann architecture or on an adiabatic quantum computer based on superconducting qubits. Memcomputing machines could be implemented on self-organizing memristor-based circuits.

In detail, Memcomputing machines [21] are electronic circuits comprising memristive components arranged to compose Self Organizing Logic Gates (SOLGs) [38]. SOLGs are invertible analogues of the usual logic gates, since the input values are inferred from the imposed output through the system dynamic. Thus, Memcomputing machines are analog circuits, with digital readout, able to invert boolean functions and solve ILP problems [39]. Memcomputing deals with ILP problems through specific circuital elements, dubbed Self-Organizing Algebraic Gates (SOAGs). SOAGs [39] share the same properties of SOLGs but their circuit is designed to self-organize to satisfy an algebraic relation as shown in Figure 2. Using SOAGs, one can assemble a Self-Organizing Algebraic Circuit (SOAC). The SOAC collectively self-organizes in order to satisfy the constraints of an ILP problem. Indeed the linear equalities and inequalities of a given ILP problem can be directly mapped on a SOAC (see Figure 2). Instead, the cost function can be easily reformulated as an extra linear inequality with an extra bounding parameter. Iteratively, this bound is reduced forcing the SOAC to self-organize and find a new feasible solution, each time closer to the global optimum. Currently no physical implementation of such concept exists, but Memcomputing Inc. has realized software able to simulate the circuital dynamic of Memcomputing machines on classical computers. The simulation software exploits GPUs to increase the overall performance.

2) D-WAVE

The D-Wave machinery belongs to the category of quantum annealing processors, also called Adiabatic Quantum Computers (AQCs). Such devices are composed by few thousands qubits that can be put in a quantum superposition state. The system evolves according to the following Hamiltonian:

$$H(t) = -F(t)\left(\sum_{i,j} J_{ij}\sigma_i^z\sigma_j^z + \sum_i h_i\sigma_i^z\right) - G(t)\sum_i \sigma_i^x \equiv F(t)H_P + G(t)H_T \ . \quad (1)$$

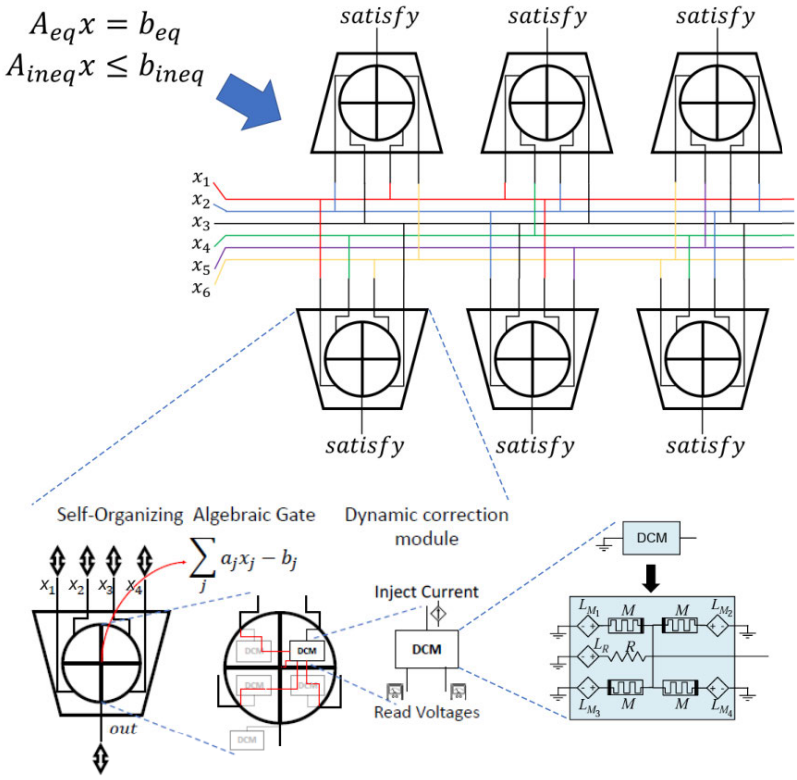

FIGURE 2. A Self-Organizing Algebraic Circuit (SOAC) represents an ILP problem. Each Self-Organizing Algebraic Gate (SOAG) is a linear condition that has to be satisfied when solving the ILP. The output of the SOAGs is imposed in order to obtain feasible solutions. The cost function is mapped into an additional SOAG whose inequality value is progressively reduced. The SOAGs at the circuital level are composed by dynamic correction modules (DCMs); the circuit components of a DCM are illustrated in the figure below on the right.

The Hamiltonian H_P implements the cost function of a QUBO problem, hence the sigma variables are binary. The Hamiltonian H_T allows for spin flips, serving as the quantum analogue for the temperature in the simulated annealing approach. By gradually reducing $G(t)$ and increasing $F(t)$, the system evolves towards the minimum of the cost function of the QUBO problem. The evolution of the Hamiltonian must take place adiabatically to guarantee success [40], [41]. In practice, thermal effects and other sources of noise render the adiabatic assumption only partially fullfilled [42], [43].

B. BENCHMARK PROBLEMS

1) SEMIPRIME FACTORIZATION PROBLEM (FP)

FP is a notoriously hard problem [44]. Given M, the number to be factored, one must find the prime factors p and q. The best-known classical algorithm for solving FP is the General Number Field Sieve whose complexity is not polynomial [45]:

$$\mathcal{O}\left(e^{\sqrt[3]{\frac{64}{9}}(\log M)^{1/3}(\log\log M)^{2/3}}\right) \tag{2}$$

In 1994, Shor proposed an algorithm [46] that solves FP in polynomial time with a gate-based quantum computer. Despite the method's correctness, its application is currently hampered by the limited number of qubits available in gate-based quantum machines. FP has high practical relevance because its prohibitive computational complexity forms the basis of the RSA [29] cryptographic system's security.

To implement this problem on the examinated platforms, we formulated FP as an optimization problem. As shown in [47], FP can be converted into a satisfability problem (SAT) problem. The first step is to convert the equation $M = p \times q$ into a nonlinear system of equations, whose unknowns are the fixed point bit representation of the numbers. The nonlinear system is then converted to a linear system with constraints, where auxiliary variables are used with corresponding restraints. This form is an integer linear programming problem and can be run on VMM and Gurobi. To allow porting on D-Wave, we defined a QUBO problem where we recover the *quadratic form* by squaring the system of equations. The constraints can be imposed via a penalty term. Full details are reported in the SM.

2) HARD–ASSIGNMENT GROMOV–WASSERSTEIN PROBLEM (GWP)

Optimal transport theory deals with the problem of moving mass from one place to another with minimal effort. This

effort is accounted by a cost function; the integral of these mass moves has to be minimized [48]. The main constraint here is represented by mass conservation; this leads naturally to approach optimal transport as a mapping problem between probability distributions. The original problem has two main formulations: the hard formulation from Monge and the relaxed formulation from Kantorovich [48]. In both, one assumes that the two metric spaces involved are the same. The Gromov–Wasserstein distance was introduced by Mémoli [30] and it is an instance of optimal transport between metric spaces having different dimensions (nonregistered) [49]. The GWP finds application in generative machine learning [50], [51] and computer graphics [30], [31], among others. Finding this distance is equivalent to finding a permutation matrix that allows this mapping between distributions.

In literature one can find several regularized, approximate or simplified forms of the GWP: Authors in [52] introduce the entropic regularization approach and uses it in combination with the Sinkhorn's matrix scaling algorithm for solving the GW problem; another paper [53] considers a variant of the optimal transport problem that restricts the set of admissible couplings to those having a low-rank factorization, achieving a linear time approximation for GWP. We will instead consider the problem's *hard–assignment* version, where the desired mapping between points is bijective and the probability distributions in the two spaces assign equal weight to all points. In this form, the cost function is an instance of a Quadratic Assignment Problem (QAP) [32], which makes it an NP-hard problem in general [54].

As anticipated, one is concerned with mapping two point clouds which belong to different metric spaces. One is given two sets of N points, $S_1 = \{x_1, \ldots, x_N\}$ and $S_2 = \{y_1, \ldots, y_N\}$, belonging to two distinct vector spaces, each equipped with a distance, $a_{ij} = d_1(x_i, x_j)$ and $b_{hk} = d_2(y_h, y_k)$. Further, one can define a distance between two pairs of points. Here, we used the squared euclidean metric $d(a_{ij}, b_{hk}) = (a_{ij} - b_{hk})^2$. We ultimately want to find the permutation matrix γ, such that the following expression is minimized:

$$\mathcal{P}_{\mathrm{GW}}(\gamma) = \sum_{ij}\sum_{hk} d(a_{ij}, b_{hk})\gamma_{ih}\gamma_{jk} \ . \tag{3}$$

Since γ is a permutation matrix, the following expressions must hold:

$$\begin{aligned} &\gamma_{ij} \in 0, 1 \quad \forall i, j \\ &\sum_i \gamma_{ij} = 1 \quad \forall j \\ &\sum_j \gamma_{ij} = 1 \quad \forall i \end{aligned} \tag{4}$$

The Gromov–Wasserstein distance is the value attained at the minimum. The implementation therefore uses n^2 binary variables. This problem is already naturally in a quadratic form similar to QUBO, yet is constrained. To ensure γ is a permutation matrix we add the following restraints to the cost function via a penalization technique (see Supplementary Material for details).

$$\begin{aligned} &\left(\sum_i \gamma_{ij} - 1\right)^2 \quad \forall j \\ &\left(\sum_j \gamma_{ij} - 1\right)^2 \quad \forall i \end{aligned} \tag{5}$$

For VMM and Gurobi, we transform this problem into a corresponding ILP by merging the products in γ into a single auxiliary variable. All details are reported in the SM.

3) CAPACITATED HELICOPTER ROUTING PROBLEM (CHRP)

CHRP is a helicopter flight scheduling problem, in which one aims to minimize the overall flight time. CHRP was introduced in [55], based on the formulation of the Dial-a-ride problem (DARP) [56]. Each flight can have multiple legs to connect offshore oil rigs. The flights are scheduled to transport workers from heliport to rigs, from rig to rig, and from rig to heliport. The problem has limiting constraints, such as the maximum range for each helicopter type and maximum capacities for the weight of the workers and luggage. As a hard constraint, CHRP requires all workers to be transported. CHRP is then a multi-agent routing problem where agents (helicopters) interact through the temporal worker assignment constraints. These characteristics make CHRP very hard, even for small instances, and therefore intractable for real world scenarios. Indeed, CHRP and similar problems such as the DARP are NP-hard in the strong sense [57] since they generalize the Travelling Salesman problem with time windows, which is proven to be NP-complete [58].

Because of its hardness and commercial relevance, multiple heuristics have been developed to provide approximate solutions, using clustering search [59], genetic algorithms [60], and a League Championship Algorithm [61]. CHRP can be cast to an integer linear programming problem with all variables being binary. The problem is then automatically in QUBO form with a null quadratic term and thus usable on D-Wave. The problem size can scale with the number of rigs (locations) and the number of workers, while the maximum number of flights is usually set according to the number of workers. In the Integer Linear Programming format, all binary variables are decision variables. The formulation has two blocks of constraints. One block defines the helicopter routes as cyclic paths in a dynamic graph. The other block defines the assignment of workers to flights. More detailed information and a description of the cost function to optimize can be found in the SM.

C. MEAN FIRST SOLUTION TIME (MFST)

It is customary, when running algorithms, to define a maximal execution time, after which the computation is stopped and a new parameter (e.g. seed) is used to run the computation. This requires proper metrics to evaluate the overall expected execution time. In particular, the estimation should include

the time spent in failures and not just the average time of successes. The estimated execution time then correlates with the allocated computing time and hence with the monetary budget for the computation. We therefore introduce a new metric, which takes fully into account the time spent in failed runs. We call this metric the Mean First Solution Time (MFST) (as it is inspired by the Mean First Passage Time concept in physics [62]).

Given a problem instance, the MFST is defined as:

$$T_{MFST} = \mathbb{E}\{k\}T_{max} + \mathbb{E}\{t\} \tag{6}$$

where $\mathbb{E}\{k\}$ is the expected number of failures before the first solution is found, T_{max} is the maximal allowed execution time (e.g. for one seed) it is not a random variable, and $\mathbb{E}\{t\}$ is the expected solution time (the averaged time of solved instances).

The expected solution time $\mathbb{E}\{t\}$ can be easily estimated via the usual sample mean estimator:

$$\bar{t}_s = \frac{1}{|I_s|}\sum_{j\in I_s} t_j \tag{7}$$

where I_s is the set of solved instances and $|I_s|$ its cardinality.

The variable k is a random variable which follows the negative hypergeometric distribution NHG($|I|, |I| - |I_s|, 1$)

$$k \sim \mathrm{NHG}(|I|, |I| - |I_s|, 1) = \frac{\binom{|I|-k-1}{|I|-|I_s|-k}}{\binom{|I|}{|I|-|I_s|}} \tag{8}$$

where $|I|$ is the cardinality of the run set. Therefore the expected value of the number of failures k before a success is:

$$\frac{|I| - |I_s|}{|I_s| + 1} \tag{9}$$

Hence, we estimate the MFST via the formula:

$$\overline{T}_{MFST} = \frac{|I| - |I_s|}{|I_s| + 1}T_{max} + \frac{1}{|I_s|}\sum_{j\in I_s} t_j \tag{10}$$

Clearly, if all instances are solved, the mean solution time is obtained.

Another widely used metric for evaluating the scalability is the time to solution scaled by the solution probability (TTS). While this quantity is formally a time, it does not explicitly consider the maximal execution time, which is captured by the MFST. In particular the Time To Solution (TTS) is defined as:

$$T_{TTS} = \frac{\mathbb{E}\{t\}}{p} \tag{11}$$

where p is the solution probability; this can be estimated as $\bar{p} = |I_s|/|I|$, hence the estimator:

$$\overline{T}_{TTS} = \frac{\bar{t}_s}{\bar{p}} \tag{12}$$

It is easy to show that there is a simple relation between $\overline{T}_{TTS}$ and $\overline{T}_{MFST}$:

$$\overline{T}_{MFST} = \frac{|I| - |I_s|}{|I_s| + 1}T_{max} + \frac{|I_s|}{|I|}\overline{T}_{TTS} \tag{13}$$

This relation shows that, if all problems are solved ($|I_s|$=$|I|$), the two metrics are the same. However when not all instances are solved, the presence of T_{max} creates a discrepancy. If T_{max} is high, the *TTS* can significantly underestimate the real execution time. In the SM, we derive the variance of the estimators.

III. RESULTS

We challenged the platforms on the above-mentioned orthogonal hard problems: Semiprime Factorization probem (FP), Hard-Assignment Gromov-Wasserstein problem (GWP) and the Capacitated Helicopter Routing Problem (CHRP).

To provide a fair comparison we take advantage of the previously introduced Mean First Solution Time (MFST). We perform a scalability assessment, analyze the parameters dependency of the solvers and the time required to deliver a first approximate solution.

A. SCALABILITY ASSESSMENT

Here, we discuss the scalability of the analyzed platforms in terms of MFST (see Methods for further details). This metric allows one to precisely capture the expected waiting time to obtain the first solution, or in other words the expected required computing time in an operative scenario. This is achieved by taking into account explicitly failed runs (no solutions found in the maximum allowed wall time). The timeout for D-Wave and VMM were set based on the available computing budget for each machine. D-Wave was run for dozens of seconds for each problem instance coherently with the granted computational time. The VMM was executed for a few hours while running the GPU backend, and for several days when utilizing the CPU backend (see each problem section for precise timeout values). Gurobi was granted a timeout of 72 hours (maximum wall time of the IIT HPC infrastructure). For every problem, Gurobi was run on a cluster node with 32 cores (2 physical sockets).

In the Supplementary Material, the results for the time to solution (TTS) are also reported. To evaluate the scalability and remove possible biases, we define n instances of a problem given a prescribed problem size N. We estimate the MFST of each problem and report the average MFSTs between all the problem instances given one single size. The scaling curve is the set of problems averaged MFSTs and results are reported in log-log scale.

1) SEMIPRIME FACTORIZATION PROBLEM (FP)

We defined $n = 5$ different problem instances (five different p, q pairs). To run the benchmark on Gurobi, we randomized 5 times the seed for each problem instance, for a total of

25 runs per problem size. In Figure 3a, a plot with the average MFSTs of the performed runs is shown. Each point is the average MFST of the problem instances belonging to the same bit size N.

The average MFST for VMM was estimated based on the same 5 instances and 120 different seeds for each instance. This setting was required to better estimate the solution probability. We used the standard CPU backend of the Memcomputing Software As a Service platform (Saas). For each run, VMM simulated 2 replicas of the circuit corresponding to FP, with a maximum timeout of 10 hours. To solve the problem, VMM usually runs a Monte Carlo algorithm to explore the space of the circuit parameters before simulating it. For this specific problem, no parameter space exploration was performed and the number of Markov chains was set to 120 to perform the calculation of the 120 different seeds in parallel on 60 cores. Hence, all the runs correspond to the same single circuit topology and parameter set.

We analyzed the scaling with respect to the increasing problem size, namely the bit size in the interval $[14, 64]$. Gurobi was very effective in relative terms (see MFST in Figure 3), since it performed a quick presolve of about half of the instances under consideration. Gurobi's ability to reduce the number of variables and constraints of the optimization problem depends on the instance and thus on the semiprime to be factorized. For $N = 64$, Gurobi performed a presolve calculation effectively enough to reduce solution time by four orders of magnitude compared to the problems where this simplification was not possible. At $N = 68$, the problems that Gurobi could not simplify exceeded the wall time of 72 hours of computation. For this reason, problems with a bit size greater than 64 were not considered. In the range $N = 37$ to $N = 64$, the MFST for Gurobi scales with a slope of 16.91 ± 0.92 (see Figure 3b).

All instances between $N = 15$ and $N = 42$ were solved at least once by VMM. The linear fit of the MFSTs in the range between 37 and 42 bits resulted in a scaling with a slope of 14.64 ± 0.62. Therefore, VMM obtained a slightly better slope with respect to Gurobi, but the overall execution time was still superior, and VMM could not solve all instances for larger bit sizes.

We also used D-Wave devices to solve FP in the interval between 14 and 17 bits. The number of device queries (which can be considered as the number of different seeds in a quantum device) was 10^4 up to $N = 14$ and then was gradually increased up to 8×10^4 at $N = 17$ ($\sim$ 12 seconds of computational time). The number of runs was doubled each time that N increased by a unit. We found no significant differences in the slope between D-Wave Advantage and the D-Wave 2000Q, but D-Wave Advantage proved slightly faster, probably because its greater connectivity allows for shorter physical qubit chains [63].

Overall, we found that D-Wave devices could only tackle small instances of the problems, whereas VMM and Gurobi could deal with significantly higher bit sizes. Gurobi was the fastest approach overall.

2) HARD-ASSIGNMENT GROMOV-WASSERSTEIN PROBLEM (GWP)

GWP's size and complexity depends on the variable N i.e. the number of points to match between the two sets (see Methods). To run our benchmark, we defined 5 problem instances for each problem size. For each instance, multiple runs were performed with different random seeds for the solvers. Results are reported in Figure 3b. For Gurobi, each instance was solved 5 times with different random seeds, for a total of 25 runs for each problem size.

For VMM, each instance was solved 5 times with different random seeds for the solver, for a total of 25 runs for each problem size, as for Gurobi. Contrary to FP and CHRP, we found an advantage in using the GPU backend of the Memcomputing Saas solver. We thus used GPUs to solve GWP on VMM, setting the timeout for each instance to $T_{max} = 1900$ seconds.

For D-Wave Advantage, every instance was sampled with thousands of annealing cycles, going from $50,000$ samples for $N = 3, 4, 5$ up to $390,000$ samples when $N = 7, 8$. We report that $390,000$ samples resulted in $\sim$ 60 seconds of access time for D-Wave Advantage.

Gurobi solved all the runs for each instance of each problem size up to $N = 16$ points. At $N = 17$, the most computationally intensive instances required a wall time exceeding 72 hours, so we solved every instance only once (a single seed). The corresponding point is indicated with a $'+'$ marker on the plot. The slope for Gurobi was 16.89 ± 0.83.

Figure 3b shows that the VMM achieved a better overall scaling than Gurobi. The slope of the fit in log-log scale was 5.89 ± 0.50, which was significantly better than the competitors. At $N = 12$ and $N = 13$, Gurobi and VMM required a similar time to solve the problem, but the wall times quickly diverged for bigger problem sizes. At $N = 16$, the VMM required an average MFST of $409 \pm 107s$ ($\sim$ 7 minutes) compared to the $9560 \pm 3730s$ required by Gurobi, i.e. VMM was $\sim$ 23 times faster than Gurobi. VMM was able to tackle problems almost up to the same size as Gurobi, but did not solve every instance of the hardest problem size, $N = 17$.

D-Wave could only tackle small instances ($N \leq 8$), thus it is hard to make a sound comparison with other technologies. At $N = 8$, D-Wave Advantage solved only one of the five instances, finding the correct solution for this instance only once over $390,000$ trials. The resulting point (the rightmost in the D-Wave plot) is thus of limited statistical significance. Deeper insights will be achieved when the D-Wave hardware is advanced enough to tackle bigger instances of this problem.

Overall, VMM was slightly less reliable than Gurobi in the biggest problem sizes, but showed the best scaling behaviour.

3) CAPACITATED HELICOPTER ROUTING PROBLEM (CHRP)

We fixed the number of rigs to 25 and varied the number of workers (w) and the pick-up and drop-off locations generated at random (see Methods). When converted into QUBO and embedded into chimera topology, the problem becomes too

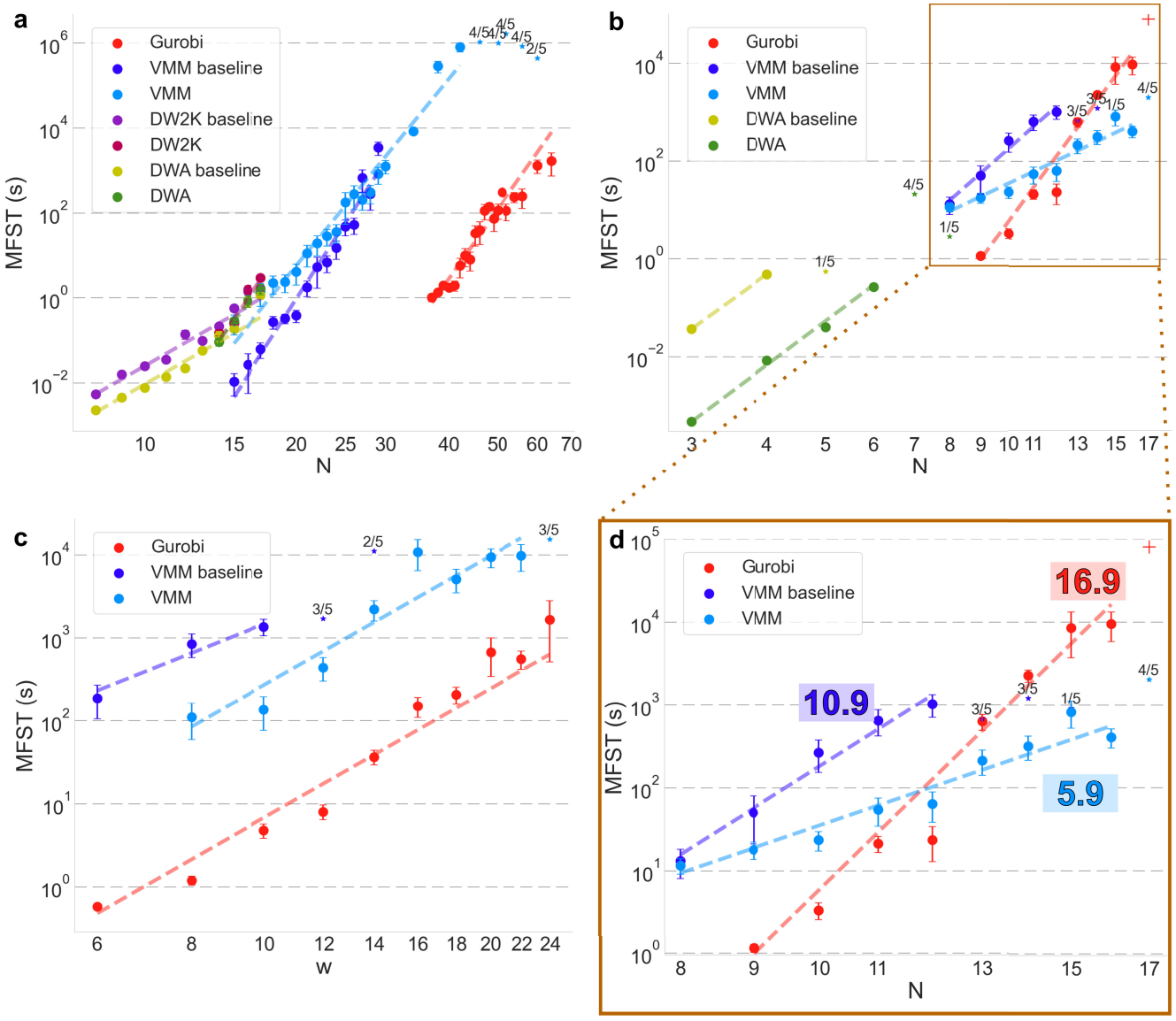

FIGURE 3. MFST plots for all the tested computing platforms, in log_{10}, log_{10} scale. Every problem size corresponds to 5 different problem instances using different seeds. Whenever a problem size on a given machine includes unsolved instances, a smaller dot is used and the number of solved instances is shown. The error bars represent the standard deviation (see Methods). We report both baseline and parameter-optimized scaling results. In the scaling section, we discuss these results. **a:** FP MFST with respect to the number of bits of the semiprime. **b:** GWP MFST with respect to the number of points. The red plus mark is the point $N = 17$ for Gurobi, which was obtained by solving each instance only once, due to time constraints. **c:** CHRP MFST with respect to the number of workers. **d:** Zoom of the GWP plot showing the slopes for Gurobi and VMM solvers for the biggest problems. The VMM with enhanced settings achieved the best performances. The highlighted rounded slope values have the following values and standard deviations: Gurobi 16.89 ± 0.83; VMM 5.89 ± 0.50; VMM baseline 10.93 ± 0.98.

large to fit in D-Wave, even for a few passengers, thus the device was not included in the comparison plot.

Summary results are presented in Figure 3c. We considered five instances for each problem size, and every instance was submitted to Gurobi and VMM with 5 different seeds, for a total of 25 submitted problems for every worker size. We used the GPU backend of the Memcomputing Saas, setting the timeout for each instance to 5 hours. The slopes of the two solvers in log-log scale were very similar for the problem sizes considered (5.2 for Gurobi versus 5.1 for VMM). In contrast to GWP, the CHRP did not show an advantage for VMM in scaling terms. Additionally, while Gurobi solved all instances, VMM was slightly less reliable beacause it could not solve all instances of the biggest size ($w = 24$).

Overall, the results of the three benchmark problems show that D-Wave's current hardware can solve only very small instances. VMM managed the biggest problems well in most cases. VMM's scaling was similar to Gurobi, but was superior for one problem.

We note that, for GWP, VMM used GPU hardware to achieve significant speed-ups. In contrast, Gurobi, and the branch-and-bound [64] method in general, is not particularly

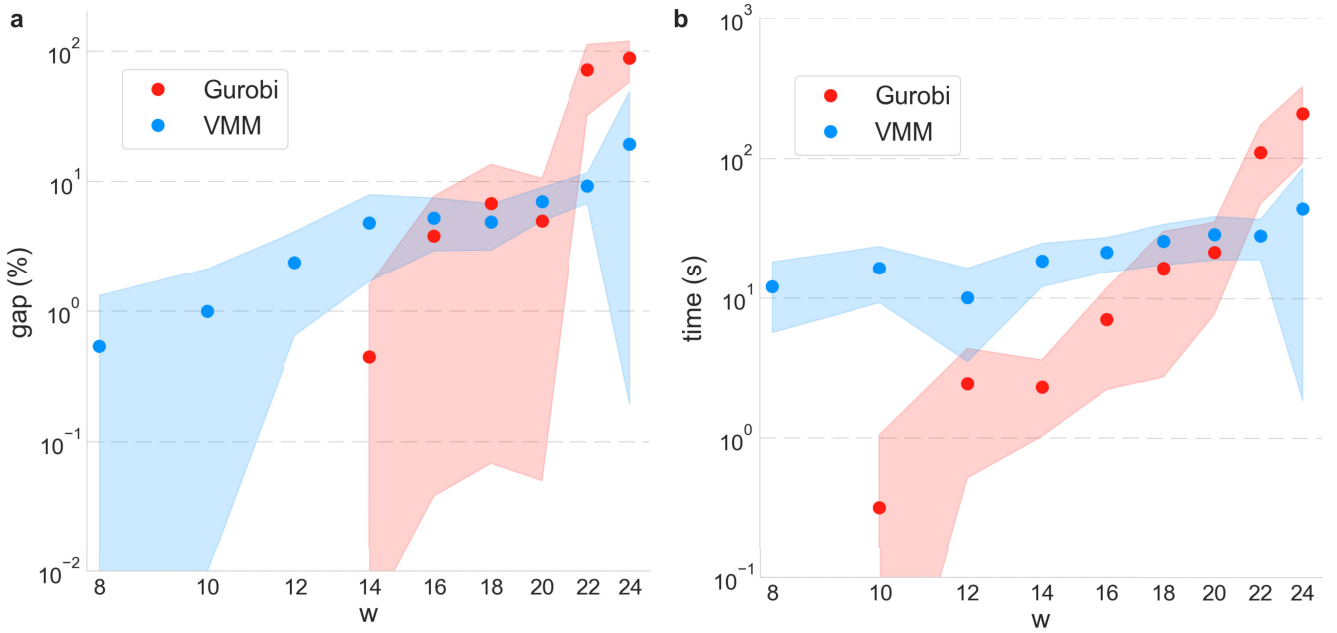

FIGURE 4. Performances of VMM and Gurobi on the CHRP problem, in log-log scale. Errorbars have been slipped whenever they included negative values. a: average gap percentage with respect to the optimal solution reached by VMM and Gurobi in 60 seconds, versus the number of workers. Gurobi was faster for small instances but its performances quickly deteriorated, while VMM was much more solid as the problem size increased. b: time required by VMM and Gurobi to reach the first feasible solution, versus the number of workers. The dependence of computing time on problem size clearly differed between the two solvers. VMM was more resilient to hard instances, resulting in a better wall time for $w = 22, 24$.

amenable to a GPU implementation. This is important because the ability to exploit GPU architectures may be critical to improving the overall computing time.

B. PARAMETERS DEPENDENCY

D-Wave and VMM allow users to tune several parameters that directly affect the dynamic of the physical and simulated device, respectively. For each machine, we identified a single problem size on which to execute a tuning protocol. Those optimal parameters were then used for all the other problem sizes, improving performances. Below, we systematically compare the scaling results using the default parameters against their tuned version. The parameter tuning procedure is described in detail in the SM.

1) SEMIPRIME FACTORIZATION PROBLEM (FP)

For VMM, we considered $N = 29$ as the tuning point. In Figure 3a, a comparison between the results before and after the parameter tuning is shown. Instances $N \in [30, 42]$ were not solved using default parameters. After the parameter tuning, all the instances up to $N = 42$ were solved in less than $\sim 4 \times 10^3 s$. For D-Wave, the size $N = 14$ was used as the tuning point. The tuned parameters didn't result in an improved solution probability.

2) HARD–ASSIGNMENT GROMOV-WASSERSTEIN PROBLEM (GWP)

In the GWP, VMM without any parameter tuning solved only one of the five instances for $N = 15$, with no instances solved for $N = 16$ and $N = 17$ (see Figure 3b). The parameter tuning was performed on one instance at $N = 16$, enabling VMM to solve all other instances for $N = 16$ and all but one instance for $N = 17$. The tuning process also sensibly reduced the computing time for all $N < 16$ cases. This means that for GWP the parameter tuning of VMM shows good transferability: spending computational time to get the right parameters is an effort that systematically boosts the solver's ability to tackle new GWP instances, even at different problem sizes. The most remarkable effect was at $N = 12$, where the MFST of VMM was reduced by a factor of 16.

The effect for D-Wave Advantage was even more remarkable: the tuning procedure at $N = 6$ reduced the MFST by more than one order of magnitude at all problem sizes. The highest speed-up ($\sim$ 78 times faster) was found at $N = 3$.

3) CAPACITATED HELICOPTER ROUTING PROBLEM (CHRP)

Parameter tuning allowed VMM to reduce all its MFSTs, peaking to a tenfold reduction for $w = 10$ (see Figure 3c). While the baseline VMM solved all instances up to $w = 10$, VMM with tuned parameters solved all instances up to $w = 22$. The tuned VMM could tackle problems with $\sim$ 3.5 times the number of nonzero elements in the ILP problem matrix, compared to the baseline. The efficacy of the solver was therefore greatly increased, showing good transferability of the optimal parameters.

Overall, we conclude that the tested parameter tuning procedures are fundamental for VMM and D-Wave solvers. For some instances of GWP and CHRP, using optimal parameters

reduced by more than one order of magnitude the required time to find the global minimum. The ability to obtain an advantage using new computational approaches such as VMM and D-Wave heavily depends on the development of better and automated parameter tuning procedures.

C. GAP AND LATENCY DRIVEN ANALYSIS

CHRP finds direct application in industry. The ability to quickly reach a good solution, i.e. latency, is more relevant than being able to reach the global optimum in several real-world scenarios. Hence, we analyzed the solver's ability to achieve a certain gap from the true solution in a given amount of time. Figure 4a shows the gap from the optimal solution obtained by the two solvers with one minute as maximum wall time. When the solver did not find a feasible solution, the gap was considered 100%. For a number of workers $w = 22, 24$, Gurobi struggled to find a feasible solution in a minute, resulting in gap values up to $89 \pm 31\%$ at $w = 24$. VMM achieved better scores on the biggest sizes, with a $19 \pm 30\%$ gap at $w = 24$. The gap obtained by both solvers exhibits great variability (as can be seen by the standard deviation) due to the different difficulty of the tackled instances.

In Figure 4b, we report the time required by the two solvers to find the first feasible solution. When w increases, Gurobi takes a rapidly increasing amount of time to satisfy the constraints. However, VMM finds a feasible solution in a time that seems only mildly dependent on the problem size. For comparison, Gurobi takes 0.32 ± 0.73 seconds for $w = 10$ (first nontrivial case for the solver) and 208 ± 116 seconds for $w = 24$ (i.e. $\sim$ 650 times slower due to increased problem complexity). VMM takes 16.5 ± 7.2 seconds for $w = 10$, and 44 ± 42 seconds for $w = 24$, which is only $\sim$ 2.7 times slower on average.

IV. DISCUSSION AND CONCLUSIONS

Here, we challenged two novel optimization engines on three hard problems and compared them with the state-of-the-art classical Turing Gurobi ILP solver. Gurobi was fast and reliable for all the tested problems. However, the non-classical solver VMM significantly outperformed Gurobi in absolute MFST and scaling for GWP. This finding shows that the VMM solver is very effective in some Integer Linear Programming classes. Yet, this performance boost was not effortless, being obtained only after parameter tuning. However, the parameter tuning, as for D-Wave, was performed for only one instance and the same parameters were used to solve the other ones. It is also worth mentioning that different parameter settings does not just lower the absolute convergence time, but also lower the slope of the scaling. VMM and D-Wave parameters could have been fine-tuned at different problem sizes and this could have significantly improved the scaling results. However, computing time limitations prevented this per-problem optimization and hence reaching larger problem sizes. In principle VMM has no intrinsic limits in dealing with bigger problem sizes; however, being a heuristic approach its efficiency depends on the parameters setting. A key factor is represented by the invariance of the problem structure at increasing problem sizes. When the invariance is preserved, good parameters at a given size become ''portable'' to bigger problems. If the structure invariance is not preserved (as in our hard problems) then this requires fine-tuning the parameters at each problem size.

On a practical level, VMM was also the best solver for CHRP to find a low gap feasible solution for the hardest instances. VMM's performance, in terms of absolute time but not scaling, is partly due to its use of GPUs. In contrast, Gurobi is bound to CPUs because it exploits branch and cut and heuristics that are not easily parallelizable. VMM, instead, is natively parallelizable, so able to exploit GPU (and other distributed hardware) computing power. The fact that an algorithm is not prone to GPU porting may significantly impact its longevity. Indeed, given the current GPUs power, even a relatively modest algorithmic improvement which favours a GPU porting may render CPU-based method rapidly obsolete and slow. This consideration is a very practical one and holds even if the algorithm improvement does not affect scalability theoretically.

In contrast to Gurobi and VMM, D-Wave quantum annealers tackled only small instances of the benchmark problems. D-Wave devices have often been tested on spin-glass models, which are not always representative of several applicative problems. For example, in [65] the authors suggest than spin-glass problems could lead to understating the performances achievable by AQCs; the study in [66] discusses how spin-glass benchmarks could advantage Simulated Annealing approaches with respect to AQCs; Authors in [33] show that industrially relevant problems are much harder in general than solving spin-glass models. By testing the device on real-world problems cast to QUBO forms, we observed that the solver's abilities are currently limited. This is also because on D-Wave one needs to use penalties to enforce constraints, which is not ideal. The main current bottlenecks for this technology is the thermal noise, but the technology would also benefit from an increase in the number of qubits and a more connected topology. Despite such shortcomings, the tuning procedure we performed is able to accelerate D-Wave performances significantly (up to 78 times for the GWP). Similar approaches could prove to be fundamental in bringing practical applications of adiabatic quantum computers closer.

VMM proved to be already today a valuable tool for some problems, also in terms of latency. Promisingly, VMM is only a circuit emulation, whereas a physical circuit would perform much faster and would be much more energy efficient albeit if possibly less flexible than the simulation. Energy efficiency is a key metric which future computing paradigms ought to carefully consider to grant a long term sustainability.

We can envision that such non Turing solvers might find the best application when used in tandem and not in isolation versus a classical counterpart [67]. We expect that a future challenge for High Performance Computing will be to get the best of the two worlds, achieving speed and maintaining energetic efficiency and programmability.

ACKNOWLEDGMENT

The authors gratefully acknowledge the High Performance Computing Support Team at the Data Science and Computation Facility at Fondazione Istituto Italiano di Tecnologia.

They would like to thank CINECA for granting them the access to 3 hours of computational time on D-Wave Systems AQCs. They also thank Memcomputing Inc. for granting them the access to several days of computational time on their CPU and GPU based Saas solver.

COMPETING INTERESTS STATEMENT

Fabio Traversa is CTO and co-founder of Memcomputing, Inc. whose Virtual Memcomputing Machine (VMM) has been employed in this manuscript.

CODE AVAILABILITY STATEMENT

D-Wave and VMM solvers are commercially available platforms. All the other support codes to instantiate the problems and run the analysis are available upon request.

DATA AVAILABILITY STATEMENT

All the input problems and results data are available upon request.

REFERENCES

[1] J. Schmidhuber, "Colossus was the first electronic digital computer," *Nature*, vol. 441, no. 7089, p. 25, May 2006.
[2] R. Herken, *The Universal Turing Machine A Half-Century Survey*. Berlin, Germany: Springer-Verlag, 1995.
[3] L. F. Menabrea, "Sketch of the analytical engine (1843) with notes by the translator, Ada Agusta, countess of Lovelace," Tech. Rep., 2021, ch. 3, pp. 9–26, doi: 10.7551/mitpress/12274.003.0005.
[4] S. E. Thompson and S. Parthasarathy, "Moore's law: The future of Si microelectronics," *Mater. Today*, vol. 9, no. 6, pp. 20–25, Jun. 2006.
[5] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge, U.K.: Cambridge Univ. Press, 2010.
[6] D. Deutsch, "Quantum theory, the church–turing principle and the universal quantum computer," *Proc. Royal Soc. A*, vol. 400, no. 1818, pp. 97–117, 1985.
[7] E. Bernstein and U. Vazirani, "Quantum complexity theory," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1411–1473, 1997.
[8] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Phys. Rev. A, Gen. Phys.*, vol. 52, no. 5, pp. 3457–3467, Nov. 1995.
[9] (2021). *Xanadu Quantum Technologies*. [Online]. Available: https://www.xanadu.ai/hardware
[10] IonQ. (2021). *Ionq–Trapped Ion Quantum Computing*. [Online]. Available: https://ionq.com/
[11] (2021). *Honeywell Quantum Solutions*. [Online]. Available: https://www.honeywell.com/
[12] (2021). *Google Quantum AI*. [Online]. Available: https://quantumai.google/hardware
[13] (2021). *IBM Quantum*. [Online]. Available: https://quantum-computing.ibm.com/
[14] (2021). *Rigetti Computing*. [Online]. Available: https://www.rigetti.com/what-we-build
[15] E. Gibney, "Quantum gold rush: The private funding pouring into quantum start-ups," *Nature*, vol. 574, no. 7776, pp. 22–24, Oct. 2019, doi: 10.1038/d41586-019-02935-4.
[16] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, and D. A. Buell, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.
[17] E. Pednault, J. Gunnels, D. Maslov, and J. Gambetta. (2019). *Quantum Supremacy*. [Online]. Available: https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/ and https://web.archive.org/web/20230215010715/
[18] Y. Liu, X. Liu, F. Li, H. Fu, Y. Yang, J. Song, P. Zhao, Z. Wang, D. Peng, H. Chen, C. Guo, H. Huang, W. Wu, and D. Chen, "Closing the 'quantum supremacy' gap: Achieving real-time simulation of a random quantum circuit using a new sunway supercomputer," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2021, pp. 1–12.
[19] F. Pan, K. Chen, and P. Zhang, "Solving the sampling problem of the sycamore quantum circuits," *Phys. Rev. Lett.*, vol. 129, no. 9, Aug. 2022, Art. no. 090502, doi: 10.1103/PhysRevLett.129.090502.
[20] M. W. Johnson, M. H. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, and P. Bunyk, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, May 2011.
[21] F. L. Traversa and M. Di Ventra, "Universal memcomputing machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 11, pp. 2702–2715, Nov. 2015.
[22] K. Tatsumura, M. Yamasaki, and H. Goto, "Scaling out Ising machines using a multi-chip architecture for simulated bifurcation," *Nature Electron.*, vol. 4, no. 3, pp. 208–217, Mar. 2021.
[23] N. Mohseni, P. L. Mcmahon, and T. Byrnes, "Ising machines as hardware solvers of combinatorial optimization problems," *Nature Rev. Phys.*, vol. 4, no. 6, pp. 363–379, May 2022, doi: 10.1038/s42254-022-00440-8.
[24] K. Y. Camsari, B. M. Sutton, and S. Datta, "P-bits for probabilistic spin logic," *Appl. Phys. Rev.*, vol. 6, no. 1, Mar. 2019, Art. no. 011305, doi: 10.1063/1.5055860.
[25] M. Kowalsky, T. Albash, I. Hen, and D. A. Lidar, "3-regular three-XORSAT planted solutions benchmark of classical and quantum heuristic optimizers," *Quantum Sci. Technol.*, vol. 7, no. 2, Feb. 2022, Art. no. 025008, doi: 10.1088/2058-9565/ac4d1b.
[26] 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*, vol. 345, no. 6195, pp. 420–424, Jul. 2014.
[27] E. Knill, R. Laflamme, R. Martinez, and C.-H. Tseng, "An algorithmic benchmark for quantum information processing," *Nature*, vol. 404, no. 6776, pp. 368–370, Mar. 2000.
[28] Gurobi Optimization and LLC. (2021). *Gurobi Optimizer Reference Manual*. [Online]. Available: https://www.gurobi.com
[29] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978, doi: 10.1145/359340.359342.
[30] F. Mémoli, "Gromov–Wasserstein distances and the metric approach to object matching," *Found. Comput. Math.*, vol. 11, no. 4, pp. 417–487, Aug. 2011.
[31] G. Peyré, M. Cuturi, and J. Solomon, "Gromov–Wasserstein averaging of kernel and distance matrices," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2664–2672.
[32] E. L. Lawler, "The quadratic assignment problem," *Manage. Sci.*, vol. 9, no. 4, pp. 586–599, 1963.
[33] A. Perdomo-Ortiz, A. Feldman, A. Ozaeta, S. V. Isakov, Z. Zhu, B. O'Gorman, H. G. Katzgraber, A. Diedrich, H. Neven, J. de Kleer, B. Lackey, and R. Biswas, "Readiness of quantum optimization machines for industrial applications," *Phys. Rev. Appl.*, vol. 12, no. 1, Jul. 2019, Art. no. 014004.
[34] I. Hen, J. Job, T. Albash, T. F. Rønnow, M. Troyer, and D. A. Lidar, "Probing for quantum speedup in spin-glass problems with planted solutions," *Phys. Rev. A, Gen. Phys.*, vol. 92, no. 4, Oct. 2015, Art. no. 042325.
[35] T. Albash and D. A. Lidar, "Demonstration of a scaling advantage for a quantum annealer over simulated annealing," *Phys. Rev. X*, vol. 8, no. 3, Jul. 2018, Art. no. 031016.
[36] S. Ghose, A. Boroumand, J. S. Kim, J. Gómez-Luna, and O. Mutlu, "Processing-in-memory: A workload-driven perspective," *IBM J. Res. Develop.*, vol. 63, no. 6, p. 3, Nov. 2019.
[37] L. O. Chua and G. N. Lin, "Non-linear optimization with constraints: A cook-book approach," *Int. J. Circuit Theory Appl.*, vol. 11, no. 2, pp. 141–159, Apr. 1983.
[38] S. R. B. Bearden, H. Manukian, F. L. Traversa, and M. D. Ventra, "Instantons in self-organizing logic gates," *Phys. Rev. Appl.*, vol. 9, no. 3, Mar. 2018, Art. no. 034029.

[39] F. L. Traversa and M. Di Ventra, "MemComputing integer linear programming," 2018, *arXiv:1808.09999*.
[40] S. Morita and H. Nishimori, "Convergence of quantum annealing with real-time Schrödinger dynamics," *J. Phys. Soc. Jpn.*, vol. 76, no. 6, Jun. 2007, Art. no. 064002.
[41] S. Morita and H. Nishimori, "Mathematical foundation of quantum annealing," *J. Math. Phys.*, vol. 49, no. 12, Dec. 2008, Art. no. 125210.
[42] M. H. Amin, "Searching for quantum speedup in quasistatic quantum annealers," *Phys. Rev. A, Gen. Phys.*, vol. 92, no. 5, Nov. 2015, Art. no. 052323.
[43] T. Albash and D. A. Lidar, "Decoherence in adiabatic quantum computation," *Phys. Rev. A, Gen. Phys.*, vol. 91, no. 6, Jun. 2015, Art. no. 062320.
[44] S. G. Krantz, *The Proof is in The Pudding: The Changing Nature of Mathematical Proof*. Cham, Switzerland: Springer, 2011.
[45] A. K. Lenstra, H. W. Lenstra, M. S. Manasse, and J. M. Pollard, "The number field sieve," in *The Development of the Number Field Sieve*. Cham, Switzerland: Springer, 1993, pp. 11–42.
[46] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, 1994, pp. 124–134.
[47] C. J. Burges, "Factoring as optimization," Microsoft Res., Redmond, WA, USA, Tech. Rep. MSR-TR-200, 2002.
[48] L. Ambrosio, L. A. Caffarelli, Y. Brenier, G. Buttazzo, C. Villani, and S. Salsa, *Optimal Transportation and Applications*. Berlin, Germany: Springer, 2003, doi: 10.1007/b12016.
[49] F. Mémoli, "Spectral Gromov–Wasserstein distances for shape matching," in *Proc. IEEE 12th Int. Conf. Comput. Vis. Workshops*, Sep. 2009, pp. 256–263.
[50] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 214–223.
[51] C. Bunne, D. Alvarez-Melis, A. Krause, and S. Jegelka, "Learning generative models across incomparable spaces," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 851–861.
[52] J. Solomon, G. Peyré, V. G. Kim, and S. Sra, "Entropic metric alignment for correspondence problems," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 1–13, Jul. 2016.
[53] M. Scetbon, G. Peyré, and M. Cuturi, "Linear-time Gromov Wasserstein distances using low rank couplings and costs," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 19347–19365.
[54] T. Vayer, R. Flamary, R. Tavenard, L. Chapel, and N. Courty, "Sliced Gromov–Wasserstein," in *Proc. Neural Inf. Process. Syst.*, 2019, pp. 1–11.
[55] M. T. Fiala Timlin and W. R. Pulleyblank, "Precedence constrained routing and helicopter scheduling: Heuristic design," *Interfaces*, vol. 22, no. 3, pp. 100–111, Jun. 1992.
[56] J.-F. Cordeau and G. Laporte, "The dial-a-ride problem (DARP): Variants, modeling issues and algorithms," *Quart. J. Belg., Fr. Italian Operations Res. Societies*, vol. 1, no. 2, pp. 89–101, Jun. 2003.
[57] R. W. Calvo and A. Colorni, "An effective and fast heuristic for the dial-a-ride problem," *4OR*, vol. 5, no. 1, pp. 61–73, Mar. 2007.
[58] M. W. P. Savelsbergh, "Local search in routing problems with time windows," *Ann. Operations Res.*, vol. 4, no. 1, pp. 285–305, Dec. 1985.
[59] R. D. A. Rosa, A. M. Machado, G. M. Ribeiro, and G. R. Mauri, "A mathematical model and a clustering search metaheuristic for planning the helicopter transportation of employees to the production platforms of oil and gas," *Comput. Ind. Eng.*, vol. 101, pp. 303–312, Nov. 2016.
[60] A. Motta, R. Vieira, and J. Soletti, "Optimal routing offshore helicopter using genetic algorithm," in *Proc. 6th IEEE Joint Int. Inf. Technol. Artif. Intell. Conf.*, vol. 2, Aug. 2011, pp. 6–9.
[61] A. H. Kashan, A. Abbasi-Pooya, and S. Karimiyan, "A rig-based formulation and a league championship algorithm for helicopter routing in offshore transportation," in *Proc. 2nd Int. Conf. Data Eng. Commun. Technol.*, A. J. Kulkarni, S. C. Satapathy, T. Kang, and A. H. Kashan, Eds. Singapore: Springer, 2019, pp. 23–38.
[62] M. Gitterman, "Mean first passage time for anomalous diffusion," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 62, no. 5, pp. 6065–6070, Nov. 2000.
[63] M. Maronese, L. Moro, L. Rocutto, and E. Prati, "Quantum compiling," 2021, *arXiv:2112.00187*.
[64] L. G. Mitten, "Branch-and-bound methods: General formulation and properties," *Operations Res.*, vol. 18, no. 1, pp. 24–34, Feb. 1970.
[65] H. G. Katzgraber, F. Hamze, and R. S. Andrist, "Glassy chimeras could be blind to quantum speedup: Designing better benchmarks for quantum annealing machines," *Phys. Rev. X*, vol. 4, no. 2, Apr. 2014, Art. no. 021008.
[66] H. G. Katzgraber, F. Hamze, Z. Zhu, A. J. Ochoa, and H. Munoz-Bauza, "Seeking quantum speedup through spin glasses: The good, the bad, and the ugly," *Phys. Rev. X*, vol. 5, no. 3, Sep. 2015, Art. no. 031026.
[67] A. Callison and N. Chancellor, "Hybrid quantum-classical algorithms in the noisy intermediate-scale quantum era and beyond," *Phys. Rev. A, Gen. Phys.*, vol. 106, no. 1, Jul. 2022, Art. no. 010101, doi: 10.1103/PhysRevA.106.010101.

LORENZO ROCUTTO (Member, IEEE) received the M.D. degree in theoretical physics from the University of Milano-Bicocca. He is currently pursuing the joint Ph.D. degree in data science and computation with the University of Bologna and the Italian Institute of Technology. His research focuses on computation with non-Von Neumann frameworks, such as adiabatic quantum processors, analog circuits, and simulated Ising machines. He has published two articles where adiabatic quantum computers are exploited to train unsupervised learning algorithms.

MARCO MARONESE (Member, IEEE) received the degree in theoretical physics from the University of Milan-Bicocca, with a thesis in quantum machine learning. He is currently pursuing the Ph.D. degree in data science and computation with the University of Bologna, Italy. He was a consultant in the field of quantum algorithms. He develops quantum algorithms for many vertical applications, especially for computational chemistry and finance, collaborating with the Italian Institute of Technology, the National Research Council of Italy, and private companies.

FABIO LORENZO TRAVERSA received the Laurea degree in nuclear engineering and the Ph.D. degree in physics from Politecnico di Torino, Torino, Italy, in 2004 and 2008, respectively. He is the Chief Technology Officer (CTO) of MemComputing, Inc. He co-founded the company in 2016 with Fellow Physicist Max Di Ventra, Ph.D., and noted San Diego-based tech entrepreneur John Beane. Prior to joining MemComputing, Inc., he was a Research Scientist with the University of California at San Diego, San Diego, CA, USA, where he specialized in unconventional computing methods with Dr. Di Ventra. Together, they invented the technology known as memcomputing. To date, they've received three patents related to the technology with more pending domestically and internationally. He was a Researcher Fellow with the Electronics Department, Politecnico di Torino, in 2008. From 2009 to 2014, he was with Departament d'Enginyeria Electrònica, Universitat Autònoma de Barcelona (UAB), Barcelona, Spain, as a Postdoctoral Researcher and then a Research Fellow. During the same period, he was also a Visiting Researcher with the University of California at San Diego, and with New York University, New York, NY, USA. From 2015 to 2016, he was a Scientist with the University of California at San Diego and in 2016, a Visiting Professor with Politecnico di Torino. Internationally, he is well known for his invited talks on electronics, physics, and unconventional computing. He has authored 80+ scientific articles published in top-tier refereed journals. His research interests cover unconventional computing, memcomputing, high-performance computing, numerical algorithms, computational complexity, circuit theory, circuit design, electronic transport in nano-devices, quantum transport, quantum computing, stability analysis of nonlinear circuits and systems, and noise analysis of nonlinear circuits. For more information, see https://scholar.google.com/citations?user=FJ8phpYAAAAJ&hl=en.

SERGIO DECHERCHI received the Laurea degree (summa cum laude) in electronic engineering from Genoa University, Italy, in 2007, and the Ph.D. degree in electronic engineering and computer science on machine learning and data mining from the Department of Biophysics and Electronic Engineering (DIBE), Genoa University, in 2011. From 2011 to 2016, he was a Postdoctoral Researcher with the Department of Drug Discovery and Development (D3), Istituto Italiano di Tecnologia (IIT), Genoa, Italy, where he designed, developed, and applied computational intelligence/chemistry methods to drug discovery. He is the Designer and Developer of NanoShaper, a tool for molecular surface computation and pocket detection. In 2014, he co-founded BiKi Technologies s.r.l., a company dealing with molecular dynamics and machine learning methods for drug discovery. From 2017 to 2022, he was a Technologist with IIT. In 2022, he obtained the national habilitation as an Associate Professor in computer science. Since 2023, he has been the Coordinator of the Data Science and Computation IIT Facility. He is the author of more than 70 papers in peer-reviewed journals and conferences in the fields of computational intelligence and computational chemistry. He has received some awards, EU/national/private grants, delivered several invited talks/lectures, and co-organized some workshops (e.g. ECAM/CECAM). He is a reviewer for EU and national funding agencies. He serves as an Associate Editor for *Cognitive Computation* (Springer).

ANDREA CAVALLI received the Ph.D. degree in pharmaceutical sciences from the University of Bologna, in 1999. He is the Director of CECAM (European Center for Atomistic and Molecular Computations) at EPFL, Lausanne. He is a Professor of medicinal chemistry with the University of Bologna and leads Computational and Chemical Biology at the Italian Institute of Technology, Genoa, Italy. He did postdoctoral work at SISSA, Trieste, Italy, and ETH, Zurich, Switzerland. He was a Visiting Professor with the University of California at San Diego, in 2019. His research interests are in the field of computational chemistry and drug discovery. In particular, he has designed, developed, and applied new algorithms and codes to accelerate the discovery of new medicines in therapeutic areas, including cancer, neurodegenerative diseases, and neglected tropical diseases. He is the author of about 300 articles and has delivered over 120 invited lectures and seminars at international congresses and at prestigious international institutions. He is a co-inventor in 12 international PCT patents and patent applications, and he is a Co-Founder of the high-tech start-up company BiKi Technologies. From 2017 to 2022, he was the Chair of the international scientific organization QSAR, Chemoinformatic and Modeling Society (QCMS, former QSAR Society). He is a member of the editorial board of numerous international scientific journals, and he has been a reviewer and a committee member for several international funding agencies (EU, Austria, Switzerland, France, USA, Israel, Norway, and Portugal).

Open Access funding provided by 'Istituto Italiano di Tecnologia' within the CRUI CARE Agreement