TIGER: Topology-aware Task Assignment Approach using Ising machines

Anastasiia Butko

Lawrence Berkeley National Laboratory, Berkeley, CA
abutko@lbl.gov

David Donofrio

Lawrence Berkeley National Laboratory, Berkeley, CA
ddonofrio@lbl.gov

George Michelogiannakis

Lawrence Berkeley National Laboratory, Berkeley, CA
mihelog@lbl.gov

John Shalf

Lawrence Berkeley National Laboratory, Berkeley, CA
jshalf@lbl.gov

ABSTRACT

Optimal mapping of a parallel code's communication graph is increasingly important as both system size and heterogeneity increase. However, the topology-aware task assignment problem is an NP-complete graph isomorphism problem. Existing task scheduling approaches are either heuristic or based on physical optimization algorithms, providing different speed and solution quality trade-offs. Ising machines such as quantum and digital annealers have recently become available offering an alternative hardware solution to solve certain types of optimization problems. We propose an algorithm that allows expressing the problem for such machines and a domain specific partition strategy that enables to solve larger scale problems. TIGER - topology-aware task assignment mapper tool - implements the proposed algorithm and automatically integrates task- communication graph and an architecture graph into the quantum software environment. We use D-Wave's quantum annealer to demonstrate the solving algorithm and evaluate the proposed tool flow in terms of performance, partition efficiency and solution quality. Results show significant speed-up of the tool flow and reliable solution quality while using TIGER together with the proposed partition.

CCS CONCEPTS

• **Hardware** → Quantum technologies; • **Software and its engineering** → Context specific languages; Development frameworks and environments;

KEYWORDS

Topology-aware task assignment, optimization, Ising machine, quantum annealing.

ACM Reference Format:

Anastasiia Butko, George Michelogiannakis, David Donofrio, and John Shalf. 2019. TIGER: Topology-aware Task Assignment Approach, using Ising machines. In *Proceedings of the 16th conference on Computing Frontiers (CF '19)*, April 30-May 2, 2019, Alghero, Italy. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3310273.3321556>

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of the United States government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

CF '19, April 30-May 2, 2019, Alghero, Italy

© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6685-4/19/05...\$15.00
<https://doi.org/10.1145/3310273.3321556>

1 INTRODUCTION

Increasing computational performance requirements with a sustainable power budget pushes computer architecture research towards extreme heterogeneity where efficient task assignment will have a definitive impact on performance. The task assignment problem aims to maximize application performance by balancing computational load among multiple and often heterogeneous processing units while reducing the overhead of task communication. The task assignment problem has been shown to be equivalent to a graph isomorphism problem by Bokhari [1], which known to be NP-complete [9, 15]. Therefore, solvers for this problem can be heuristic [23] that tradeoff solution quality for computation speed, or physical optimization algorithms, such as simulated annealing [26], genetic techniques [18], and others. In addition, solvers can optimize computational load only, communication cost only, or a weighted combination [4, 21].

While genetic algorithms and simulated annealing are often considered as the most optimal, recently available Ising machines offer an alternative hardware solution to solve a Quadratic Unconstrained Binary Optimization (QUBO) problem. These Ising machines can be implemented using different technologies and based on various physical effects, e.g. coherent Ising machines [27], recently introduced by Fujitsu digital annealer [7] and quantum annealers designed by D-Wave Systems Inc. [11].

Quantum annealing [13] is a metaheuristic technique to find the global minimum or maximum in a discrete search space. When applied to the task assignment problem, quantum annealing typically outperforms heuristic algorithms through its quantum tunneling effect [5]. In [16] authors conclude that physical quantum annealing has better scaling compared to state-of-the-art sequential methods that include simulated annealing and quantum Monte Carlo, tailored methods that solve simplified optimization problems. Also, in [14] authors emphasize the importance of determining the application domain where quantum annealing maximizes its benefits, but this has yet to be defined.

Despite the potential benefits offered by quantum annealers combined with a growing interest in alternative solutions spread across multiple fields, practical applicability of annealing machines stays highly questionable. One of the reasons lies in physical limitations of current machines related to the relatively small size of the chip and its poor connectivity [14]. Problem sizes demonstrated in comparison studies are usually not competitive with those handled by classical solvers. Therefore, effective problem partitioning and post-processing are required to continue exploiting quantum solver

capabilities while the solution for physical limitations is sought [28]. That makes most of the near-term quantum annealing-based approaches classical-quantum hybrids.

Another obstacle of quantum annealers adoption is caused by its programming complexity. Being a specialized accelerator, a quantum annealer can be accessed via a specific software stack. The D-Wave stack consists of several abstraction levels, such as Quantum Machine Instructions (QMI), System Interface and Control (SAPI), intermediate representation, etc [20]. Yet, the highest level users are required to program D-Wave is "virtual" QUBO, where "virtual" means that the compiler takes care to map and route the problem taking into account device connectivity. Transforming a problem into QUBO format is not a trivial task. Higher level tools as well as efficient algorithms are typically required [20].

In this work, we present an algorithm that allows to express the topology-aware task assignment problem to optimize both computation and communication cost of the placement. We demonstrate Topology-aware task assignment mapper (TIGER) tool that implements the proposed approach. TIGER transforms an application's task-communication graph (TCG) and an architecture description (ARC) into the QUBO format. Multiple transformation options are available, such as qbsolv qubo [2] and qasm [19] QUBO formats. TIGER uses a quantum annealer system to solve this QUBO problem much faster and with higher quality than current heuristic methods on classical digital computers. TIGER is written in Python and uses the NetworkX package [6] to create and manipulate TCG and ARC structures. Moreover, to overcome current physical limitations of existing quantum annealers, we propose a domain-specific QUBO partitioning based on the TCG dependency levels.

We evaluate the proposed methodology integrating TIGER into the D-Wave tool flow. Our evaluation targets performance, quantum annealing sensitivity and quality of the solution, and compares the D-Wave solver to the classical TABU-search algorithms implemented in the qbsolv framework. Also, we report resource requirements for a given problem in terms of number of qubits and couplers while mapping TCGs of different sizes. To the best of our knowledge, this is a first practical demonstration of solving the topology-aware task scheduling problem using a quantum annealing machine.

Our results show that a quantum annealer together with two-level problem decomposition, i.e. TIGER and qbsolv, significantly speed up time-to-solution down to 9% of the baseline. The quality of the provided solution is improved up to 15% by using D-Wave and is not affected by the proposed partitioning approach (communication-intensive workloads) or the effect is negligible (computation-intensive workloads). Finally, we discuss the scalability of the proposed approach and demonstrate the speed-up of up to 200x when scheduling a thousand of tasks on a thousand of nodes.

We would like to extend an invitation to the community to use TIGER and then contribute back to aid tool growth. Latest updates, documentation, and support can be found online ¹.

Given the relatively small size of the evaluated quantum annealer, we leave the discussion on general competitiveness of quantum annealer against classical computing out of the scope of this paper.

¹<https://github.com/lbnlcomputerarch>

Figure 1: Quantum annealing machines trends.

Our results aim to provide useful insights on the entire tool flow including classical decomposition, domain-specific partition and QUBO solvers.

The rest of the paper is organized as follow: Section 2 provides the background on the existing task-assignment approaches and quantum annealing. Section 3 describes the proposed mapping approach and implementation, as well as their integration into the complete tool flow. Section 5 shows performance, quality, sensitivity and scalability evaluation results. Section 6 concludes the work.

2 BACKGROUND

2.1 Quantum annealing

Quantum annealing [13] is a metaheuristic technique for solving local search problems, such as finding the global minimum or maximum in a discrete search space. Quantum annealing offers potential benefits against other popular heuristic algorithms through its quantum tunneling effect. This effect allows the system to penetrate energy barriers escaping from the local minima and therefore find better solutions to the original optimization problem. In [5], authors compare the performance of physical quantum annealer (D-Wave 2X quantum annealer) to simulated annealing and quantum Monte Carlo methods executed on a classical processor.

Furthermore, authors in [16] extend Google Inc. studies by comparing quantum annealing to state-of-the-art optimization methods, introducing more sophisticated assessment metrics. Their work considers three categories of optimization methods: sequential methods that include quantum annealing, simulated annealing and quantum Monte Carlo, tailored methods that solve simplified optimization problems, and nontailored methods that are generic and thus represent the state-of-the-art. Authors conclude that physical quantum annealing has better scaling compared to other sequential optimization methods, but it concedes to tailored as well as nontailored state-of-the-art methods. Also, authors emphasize the importance of determining the application domain where quantum annealing maximizes its benefits, but this has yet to be defined. Finally, King et al. in [14] introduce a problem class that can maximize usefulness of the quantum tunneling effect. Authors again compare quantum annealers to classical solvers and demonstrate three to four orders of magnitude performance speed-up in favor of quantum annealing.

Several studies demonstrate the use of quantum annealing for task scheduling. In [24], authors introduce a hybrid quantum-classical approach to solving scheduling problems. Their framework integrates quantum annealing with classical computing into a guided tree search. Classical algorithms manage a global tree search and communicate the node search in QUBO format to the quantum

Figure 2: Task Communication Graph Assignment on Heterogeneous multi-PU system: problem mapping on QUBO.

annealer. Authors test the proposed framework on three scheduling problems, i.e. graph-coloring, mars lander task scheduling, and airport runway scheduling. Results show that the quantum annealer's output can effectively prune and guide the search process. Authors motivate their work by necessity to expand on the capabilities of current quantum annealers and do not expect quantum annealers to be competitive in the near-term against classical computing.

In our work, we address a different scheduling problem, i.e. topology-aware task scheduling. The proposed TIGER framework extends existing software environments by automatically generating and dynamically adjusting QUBO files. We evaluate the tool flow in terms of quantum solver performance, task assignment quality and discuss the potential scalability of near-term machines.

2.2 D-Wave quantum annealer

A quantum annealing machine or a quantum annealer is a hardware implementation of the adiabatic quantum computing algorithm. Quantum annealers operate on a set of qubits. A qubit is a two-state quantum-mechanical system that can carry states $|0\rangle$ and $|1\rangle$ or be in *superposition* that expresses a linear superposition of the "basis states", i.e. $|0\rangle$ and $|1\rangle$. This feature forms the key power of quantum machines, which with n qubits can be in an arbitrary superposition of up to 2^n different states simultaneously. Another inherent quantum property of qubits is *quantum entanglement* where a group of qubits is coupled to each other in such a way that the state of each qubit cannot be perceived separately, but as a whole system state instead [17].

Quantum annealers provided by D-Wave Systems Inc. have been commercially available since 2011 [11]. Figure 1 shows current trends in the number of physical qubits per chip. D-Wave quantum chips are implemented using superconducting technology and require an extreme isolated environment with a temperature close to absolute zero. A closed cycle dilution refrigerator cools the processor down to 15 mK. Therefore, while the actual quantum chip is the size of a stamp, the physical volume of the whole D-Wave system reaches $20 m^3$. However, D-Wave machines consume less than 25 kW of power, mostly for the cooling system and front-end servers [12]. In around 10 years, quantum annealing chips have reached 10^3 number of qubits, promising significant performance

improvement for certain computing problems in the near future. Physically qubits are connected to each other using a so-called Chimera topology. The smallest Chimera unit contains a complete bipartite graph of eight vertices, each of which is connected to its four neighbours inside the unit and to its two neighbours outside the unit.

Quantum annealers minimize the QUBO problem described by Equation 1. The equation describes the evolution of the time-dependent Hamiltonian [10] that aims to find low-energy states in a system of N interacting spins, i.e. qubits. In Equation 1, q_i represents qubits that take value from the set $\{0, 1\}$, h_i is a weight coefficient associated with each qubit, J_{ij} denotes the strength of the couplings between two qubits q_i and q_j and N is the number of qubits.

$$E(q_1, \dots, q_N) = \sum_{i=1}^N h_i \cdot q_i + \sum_{i<j=1}^N J_{ij} \cdot q_i \cdot q_j \quad (1)$$

D-Wave annealer architectural designs impose a number of limitations on Equation 1. Notably, chips do not support all-to-all qubit connectivity. Thus, to couple two qubits located on different sides of the Chimera grid, excessive routing through other qubits is required. That dramatically cuts the number of available qubits to be purely used for problem solving. Another limitation concerns qubit weights and coupler strengths that lie in a specific range, i.e. $[-2;2]$ and $[-1;1]$ respectively, affecting the precision of the machine.

A low-level D-Wave program is expressed in the form of Equation 1 as a list of h_i and J_{ij} with the associated to them qubit numbers. The provided solution is a list of q_i values. This program is usually referred to as *Quantum Machine Instruction* (QMI). At this level, all previously listed constraints, such as qubit connectivity, variable range as well as the number of physically available qubits, have to be taken into account. That makes D-Wave programming a challenging task. However, there are several tools to provide a certain level of abstraction by taking as input a so-called "virtual" QUBO that abstracts away the size or connectivity topology of the D-Wave system and maps the problem onto the physical hardware using different optimization techniques.

Figure 3: Binary solution interpretation: computation task assignment and communication impact.

3 MAPPING ALGORITHM

3.1 Linear assignment problem

In the task allocation context, the Linear Assignment Problem (LAP) consists of placing a set of independent tasks onto a set of Processing Units (PUs), with each assignment incurring a certain cost. The objective is to assign each task to a PU such that the total cost is minimized [3].

Figure 2(a) illustrates the transformation of the LAP to QUBO. The qubit matrix Q represents the permutation matrix X , where each qubit defines the assignment of a task to a specific PU similar to x_{ij} above. An x_{ij} value of 1 represents that task i was assigned to PU j . A weight coefficient h_i (not shown) represents the computational cost of the assignment. Since solvers in current machines find local minima, we transform positive computation costs into negative numbers to prevent the solver from giving us all-zero answers. To respect assignment constraints such as assigning one task to one qubit, we use qubit couplings and give them high penalty values such as $J_{ij} \gg |h_i|$. For example, to prevent *task 0* from being placed on multiple PUs, we couple qubits $(q_0 \cdot q_1)$, $(q_0 \cdot q_2)$, $(q_0 \cdot q_3)$, $(q_1 \cdot q_2)$, $(q_1 \cdot q_3)$ and $(q_2 \cdot q_3)$ for four qubits. Therefore, if two of these qubits have the same task assigned to them, the large penalty value will make the overall solution ineligible.

3.2 Task-communication graph assignment

Applications can be represented as a weighted directed acyclic graph, usually referred to as a Task Communication Graph (TCG). A TCG is defined as a tuple $G = (V, E)$, where $V = (v_i)$ is a set of weighted vertices with the weight representing task computational cost and $E = (e_{i,j})$ is a set of weighted edges with the weight representing inter-task communication cost. An example of TCG is shown in the upper part of Figure 2(b).

Mapping of such a TCG into QUBO differs from previously-shown LAP in three aspects. First, a TCG includes not only computation cost, but also inter-task communication cost expressed with graph edges. Second, not all tasks are assigned to PUs within the same time frame. A TCG is divided into multiple *dependency levels* each of which represents a LAP. Dependency levels (groups) are shown with red dashed lines. Third, within each dependency level, the number of independent tasks can be different compared to the number of available PUs. The QUBO mapping transformation respects each of the above three constraints.

Communication edges. Each communication edge is included into QUBO by a qubit coupling. Communication cost is defined by the coupling strength. The cost is calculated based on the edge weight and on the communication path on the architecture topology. If

both source and destination tasks are assigned to the same PU communication cost is equal to zero. This is the most favourable case if the objective is to minimize data movement. For the example in Figure 2(b), to define the edge between *task0* and *task1* we couple qubits $(q_0 \cdot q_3)$ and $(q_1 \cdot q_2)$ with the associated topology-aware communication cost and qubits $(q_0 \cdot q_2)$ and $(q_1 \cdot q_3)$ with zero communication cost. Here, cost values are converted to negative numbers similar to computation cost values. Prioritizing communication or computation costs can be formulated by adding a weight factor to bias the solver.

Dependency levels. Because of dependencies, only a certain number of tasks can be assigned to PUs in parallel. This relaxes the second assignment constrain that says that no more than one task can be placed at a PU. This constrain is valid only for tasks belonging to the same dependency group. For the example shown in Figure 2(b), *task 0* is separated from *task 1* and *task 2* with a red dashed line. Thus, we couple only qubits $(q_2 \cdot q_4)$ and $(q_3 \cdot q_5)$ with the high penalty cost to prevent placing them on the same PU, which would otherwise be a valid solution for the solver. The first assignment constrain that says that a task can not be placed on multiple PUs at the same time remains unchanged.

Level adjustments. When the number of parallel tasks exceeds the number of available computing resources, an important decision has to be taken to prioritize a set of tasks in the most efficient way. This decision is reflected in the qubit matrix, i.e. the order of columns associated to specific tasks and corresponding assignment constrain couplings. Multiple approaches exist in the field, but this study is out of the scope of this paper. Here, we apply a simple cut based on the task ID increment. Figure 2(b) illustrates the case in which *task 4* belongs to dependency level 1, but is moved to the next level. In case there are no available slots in the following group of tasks, an additional level is created.

3.3 Domain-specific TCG partition

Given the number of logical qubits together with the number of couplings and constrains per single problem, we quickly exhaust the physical capabilities of quantum machines. Therefore, an intelligent problem partition is required. There has been extensive research on graph partitioning [22]. In this context, we apply the method shown in Figure 2(c). This method divides TCG into sub-graphs (SGs) based on dependency levels. The example shown in Figure 2(c) illustrates partitioning with two and three dependency levels per sub-QUBO1/2 and sub-QUBO3 respectively. The lowest degree of granularity corresponds to one dependency level per sub-QUBO. Further division of the problem will distort the concept of optimal parallel tasks assignment. The weakness of such partitioning is that

Figure 4: Topology-aware Task-Assignment using Quantum Annealing tool flow.

task ID	[cost1]	[cost2]	[cost3]	Parameter IDs	Value
0	0	10	10	Topology	MESH
1	0	20	20	Num Rows	2
...				Num Cols	2
				Num PUs	4
				PU	0..3 1,2,2,4
				Link	0..3 2,2,2,2
				...	

edge ID	task_1	task_2	[cost]
0	0	1	2
1	0	2	2
...			

a) Application TCG file format b) Architecture ARC file format

Figure 5: Topology-aware Task Assignment problem input.

only communication edges inside a SG are regarded. Thus, multiple communication edges get excluded from the problem and are not represented in the qubit matrix. Excluded edges are labelled with red crosses in Figure 2(c). This may have a significant impact on the quality of the provided solution, especially for communication-intensive applications.

Part of the novelty of our work is improving the partition by applying an interactive previous-placement-dependent approach. This approach takes advantage of dependency level-based partitioning. Sub-QUBOs are solved one after another and each previous SG placement is used to enhance following sub-QUBOs. Our mapper extends the qubit matrix with additional virtual qubits—one per each unique source task of all excluded input edges (edges that are inputs to a SG). This qubit is associated with a specific PU because the previous task placement is already known at this point. In Figure 2(c), virtual qubits are shown as red crosses inside the sub-QUBO matrices and missed edges previously shown as crossed out are illustrated with red arrows.

Our approach guides the solver towards a better solution than is possible with heuristics alone, but does not guarantee the optimal solution because the output edges of the sub-graphs are still excluded from the problem and the future placement is not available at this point. It should also be emphasized that QUBO minimizes the sum of given costs, which are abstract positive numbers. Minimizing the sum does not guarantee that parallel execution time is also minimized, if that is determined by the slowest task.

3.4 Binary solution interpretation

Figure 3 illustrates the binary solution interpretation by mapping the example graph from Figure 2 (c) on the four-unit mesh architecture. Each block corresponds to a dependency level of the task-communication graph. It contains three illustrative components, i.e. a qubit sub-matrix with solution values, computation task placement corresponding to the solution and communication traffic based on the prior task placements. In case both source and destination tasks are placed on the same unit, the communication edge is marked as a local communication. Local communications do not contribute to the data movement component of the objective function and represent the most favourable assignment for communication cost minimization.

3.5 Computation and Communication costs

Computation and communication costs have been previously discussed as abstract positive numbers. However, the nature of the cost metric determines whether the proposed method provides an optimal solution. If the cost is based on delay and the goal of the task assignment is to minimize time, the QUBO minimization will not provide the optimal placement. This is because QUBO minimizes the sum of the placement costs in each SG and it does not guarantee that if placed in parallel task execution time is minimum. For other metrics, such as data movement, power consumption, energy, the proposed method provides an optimal solution.

4 TASK ASSIGNMENT

D-Wave programming environment: Qbsolv [2] is an open source decomposing solver that focuses on large scale problems that do not fit into physical hardware. In addition to the D-Wave annealer interface, qbsolv has an embedded classical solver that implements the tabu search algorithm [8] to minimize the QUBO problem. Qasm [19] is a quantum macro assembler that provides extra flexibility for programming. A qasm program can be run on both D-Wave annealer or on qbsolv classical solver.

Tool flow and TIGER: Figure 4 shows the tool flow for the task assignment problem. The key component of the flow is the TIGER tool. TIGER is an open-source QUBO mapper written in Python.

Figure 6: Delay-to-Solution evaluation: (I) - classical TABU-search solver w/o TIGER sQ partition, (II) - quantum DW solver w/o TIGER sQ partition and (III) - quantum DW solver with TIGER sQ partition.

It uses NetworkX python package [6] to create and manipulate TCG and ARC structures, i.e. computing the computation and communication costs taking into account architecture topology. We demonstrate TIGER using D-Wave machines.

TIGER receives two files as inputs (marked as red '1' to denote step 1), namely TCG and ARC (architecture) that describe the application's TCG and architecture respectively. These files are provided by external modelling tools, such as the COSMIC benchmark suite [25] or ProgrAMR [4] framework. The format of these files is presented in Figure 5 (a) and (b). The TCG file consists of lines of two types associated to the application tasks and edges. Task lines contain a task ID and multiple cost values each of a different type, e.g. number of integer, floating point, memory access instructions. Edge lines contain an edge ID, source and destination task IDs, and a cost value, e.g. the amount of data to be transferred between two tasks in bytes. The architecture file describes the architecture topology and its details such as number of rows and columns, number of PUs and the capabilities of each PU and link, e.g. energy cost per different types of instructions, link throughput, etc.

Using the algorithm described in Section 3, TIGER maps input TCG and ARC files into the QUBO format and generates the QMI interface file (step '2'). It supports both qasm and qubo formats and can generate a single file per problem or multiple files in case the QUBO partitioning option is chosen. If size of the problem is less than the physical limit value, i.e. qubit sub-matrix size, QUBO or sub-QUBO can be directly solved (step '3'). Otherwise, it has to be further decomposed by qbsolv and then solved (step '4'). In both cases the problem is solved by two available solvers: the D-Wave annealer or a TABU search qbsolv implementation (step '5').

Finally, the solver generates mapping solutions that are sent back to the TIGER tool. If the solution corresponds to sub-QUBO (step '7'), it is used by TIGER to generate the next sub-QUBO as described in Section 3.2. If the solution is complete (step '6') or the last sub-QUBO problem is solved, TIGER calculates the final cost of the assignment through its Mapping-to-Metric (MtoM) interface (step '8'). This cost can be used to estimate the quality of the solution.

5 RESULTS

Experiments are conducted on a hybrid classical-quantum system that consists of an Intel Core i7 running at 3.3 GHz with 16 GB 2133 MHz LPDDR3, and a D-Wave 2X (DW2X) quantum annealer [11] that has 1152 qubits and 2400 couplers. We use three workload TCGs

from the COSMIC benchmark set [25] that differ in their problem size, number of tasks and edges: (a) Ultrasound-9x5x10 with 15 tasks and 15 edges, (b) Reed-Solomon (RS) Encoder-32x28x8 with 141 tasks and 140 edges and (c) Reed-Solomon Decoder-32x28x8 with 526 tasks and 789 edges.

5.1 Tool flow evaluation

For each workload we evaluate three scenarios: (I) TIGER QUBO mapper - qbsolv decomposer/TABU-search qbsolv solver - TIGER MtoM interpreter, (II) TIGER QUBO mapper - qbsolv decomposer/DW solver - TIGER MtoM interpreter and (III) TIGER QUBO mapper/TIGER SG partitioner - qbsolv decomposer/TABU-search qbsolv solver - TIGER MtoM interpreter. For each scenario, we vary the size of the architecture to a 2x2 PUs mesh, 4x4 PUs mesh, and a 8x8 PUs mesh.

Figure 6 shows the evaluation results. Here, we report the delay normalized by the total delay of the longest scenario to provide the solution. Each delay report is also broken down to its four major components. In all cases, the longest scenario is the one fully executed on a classical computer, e.g. scenario I. In addition, we show the number of logical qubits and couplers generated by TIGER's mapper (*qubits #* and *couplers #*), the number of partitions provided by qbsolv's decomposer (*partitions #*), and the number of SGs generated by TIGER's partitioner (*tiger sQs #*). The number of qubits in scenarios I and II is equal, and it is higher in scenario III because additional qubits are required to define previous sub-QUBO placements as shown in Figure 2. Similarly, the number of couplers as well as the number of partitions in scenarios I and II are equal. It is lower in scenario III due to the optimized QUBO mapping. The number of TIGER sub-QUBOs is reported only for scenario III. In scenarios I and II this TIGER option is not applied (na).

Discussion: Performance evaluation results prove that the physical quantum annealer, i.e. DW2X, can significantly reduce the delay-to-solution phase compared to the classical qbsolv solver. For the given set of benchmarks and architecture configurations, the performance speedup of the *DECOMPOSER-SOLVER* phase varies between 1.2x and 10.2x. The major portion of this improvement is caused by the replacement of the classical solver with the quantum annealer. The average value of the DW2X access time is around 20ms. This time includes programming time, sampling time and

Figure 7: Task assignment sensitivity and quality of the solution: (DW, single) - DW w/o sQ vs. classical TABU-search w/o sQ, (qbsolv, sQ) - classical TABU-search with sQ vs. classical TABU-search w/o sQ and (DW, sQ) - DW with sQ vs. classical TABU-search w/o sQ.

post-processing time. The sampling phase consists of multiple sample batches, each of which includes annealing, readout, and additional delay that allows the quantum annealer to cool down to the initial state. The annealing time is $20\mu s$. Although QUBO is solved by physical quantum annealer, a significant amount of time associated to the problem decomposition is spent by *qbsolv DECOMPOSER*. The total *D-Wave SOLVER* phase is composed of multiple D-Wave accesses, where the number of accesses is determined by the number of partitioned calls provided by *qbsolv DECOMPOSER*. Therefore, while using the quantum annealing solver the delay-to-solution phase highly depends on the quality of the classical decomposition.

In scenario III, we evaluate the impact of the domain-specific partitioning integrated into the QUBO mapper, i.e. TIGER level partitioner. Here, reported values represent the sum of all sub-QUBO parameters concerning the total number of qubits and couplers as well as delays per phase. Results show that by applying two-level QUBO partitioning (i.e. domain-specific first and classical qbsolv second), a massive speedup in time-to-solution can be achieved. For the given set of TCGs and ARCs, the *DECOMPOSER-SOLVER* phase is reduced down to 6% compared to the baseline scenario. Such an improvement has several sources. First, TIGER partition significantly simplifies the task for *qbsolv DECOMPOSER*, which performs better on a smaller subset of qubits and coupler tasks than on a single big problem. Consequently, qbsolv generates fewer partition calls thereby reducing *D-Wave SOLVER* phase delay. This

effect is particularly noticeable for larger TCGs, where the number of partitions is reduced twice. The total number of qubits and couplers is also different compared to the baseline. By applying the minimum number of qubits possible and adjusting the level of granularity (i.e. one sub-level per sub-QUBO), we reduce the total number of couplers. These improvements are achieved at the expense of having a larger number of qubits. This increase is 12% by average compared to the baseline. On the other hand, additional partitioning can potentially impact the quality of the generated solution. This effect is evaluated in the following section.

5.2 Assignment evaluation

We evaluate the assignment quality and multiple-run sensitivity in three comparison scenarios: (i) single QUBO on quantum annealer versus classical qbsolv solver (*dw, single*), (ii) partitioned sub-QUBOs versus single QUBO assignment on classical qbsolv solver (*qbsolv, sQ*) and (iii) partitioned sub-QUBOs on quantum annealer versus single QUBO assignment on classical qbsolv solver (*DW, sQ*). Architecture configuration files represent 2x2, 4x4 and 8x8 heterogeneous PU MESH with abstract PU acceleration factor varied from 1x to 4x. Link cost is equal to 2. Figure 7 shows the difference in computation, communication and total costs for the three evaluation scenarios compared to the baseline.

Discussion: In some cases, we obtain the same solution over multiple runs. If different solutions are returned, usually variation is within 5% from the mean value. For given set of experiments, DW2X quantum solver provides solution improvements for a single QUBO

compared to the classical TABU-search solver. Results show up to 8% of computation cost improvement, up to 25% of communication cost improvement, and up to 15% of total improvement. Both qbsolv sQ and DW sQ scenarios show similar behaviour in most experiments. However, again DW2X quantum solver provides better solutions, e.g. RS-Encoder mapped on 2x2 MESH and RS-Decoder mapped on 2x2 MESH.

Qbsolv sQ and DW sQ evaluations show that dependency-level partitioning indeed can significantly impact assignment quality, namely its communication constituent. This impact increases when architecture size scales. MtoM communication difference rises from 35% to 45% and then to almost 4x for *US* TCG mapped on 2x2, 4x4 and 8x8 architectures shown in Figure 7(a). Similarly, it changes from -2% to 6% and then to 60% for *RS Encoder* TCG as shown in Figure 7(b). However, computation constituent does not deteriorate. In both TCGs, task computation costs far outweigh communication edge cost. For instance, *US* computation cost ranges between 4,510 and 3,461,112, while communication highest cost is 20, 60 and 140 for 2x2, 4x4, and 8x8 MESHes respectively. Thereby, calculated edge weights and associated qubits couplings have low impact on the total QUBO cost. Indeed, the total MtoM difference follows the computational cost behaviors, e.g. DW sQ in Figure 7(b.1) or qbsolv sQ in Figure 7(a.2). By prioritizing the edge cost versus task cost, the communication MtoM difference can be significantly reduced. In contrast, *RS Decoder* TCG is communication intensive. The computation cost varies to up to 1,880, while the communication cost reaches 14,280 for 8x8 MESH. In this case, DW sQ partition does not impact the solution quality, but improves it by up to 15%.

5.3 Scalability evaluation

We estimate how scalable is the proposed approach applying the tool-flow on a larger TCG that contains 1281 tasks and 2304 edges that corresponds to the LDPC workload [25]. We map the graph on architectures with up to 1024 processing elements. Figure 8 (right axis) shows the delay behaviour for three experiment scenarios: (I) - classical TABU-search solver without TIGER sQ partition, (II) - classical TABU-search solver with TIGER sQ partition and (III) - quantum DW solver with TIGER sQ partition. Speedup bars (left axis) represent delay improvements while using quantum solver against classical per each architecture configuration. Both axes are in logarithmic scale. Results show that DW solver consistently outperforms the classical algorithm tending to higher speedups for larger architecture sizes. The classical solver w/o TIGER sQ partition line demonstrates incomparably high latencies that points in favor of the domain-specific partition efficiency.

Discussion: The presented tool flow is composed of three major steps: QUBO generation, classical decomposition and problem solving. The efficiency and scalability of the entire flow depends on the capabilities of each of the listed steps. Nevertheless, quantum annealer characteristics, such as number of qubits and their connectivity stay the key factors to determine the requirements for classical decomposition phase. Additional domain-specific partitions can significantly lower decomposition complexity, but for practical use more advanced technologies in both, hardware and software sides are yet required.

Figure 8: TIGER approach scalability.

6 CONCLUSIONS

We have proposed an algorithm of solving the topology-aware task assignment problem on physical Ising machines in order to accelerate and improve the quality of the solution to this challenging NP-complete problem. We have implemented the proposed algorithm in the TIGER tool that transforms weighted task-communication graph and architecture graphs into an appropriate format of the Hamiltonian function taking into account both computation and communication costs. We evaluated the proposed approach using D-Wave quantum annealer. In order to overcome the existing physical limitations of the current quantum annealer, we proposed a domain-specific partitioning based on the task-communication graph dependency levels. TIGER tool was integrated into the D-Wave software stack that enabled us to apply both the proposed dependency-level partitioning and partitioning provided by the qbsolv tool in a dynamic iterative way. We demonstrate the efficiency of our system comparing it to the classical qbsolv heuristic algorithm, showing significant time-to-solution improvements using our dependency-level partitioning and potential to find better solutions. The TIGER tool is publicly available online ².

For future work, we consider three major directions:

- **Comparison to the state-of-the-art classical scheduling tools:** we plan to design a methodology to compare the hardware optimizer, i.e. Ising machine, to the heuristic software tools.
- **Use other Ising machines:** we plan to expand our study running the problem on other Ising machines, such as digital annealer [7] and coherent Ising machine [27].
- **Problem partitioning algorithms and additional constraints mapping:** we plan to evaluate additional graph partitioning algorithms and alternative problem mapping algorithms, e.g. assigning multiple tasks in one node based on the node capacity.

Also, we plan to consider problems of bigger sizes to approach realistic and practically useful scenarios.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the the U.S. Department of Energy, grant agreement n° DE-AC02-05CH11231.

²<https://github.com/lbnlcomputerarch/tiger>

REFERENCES

- [1] S. H. Bokhari. 1981. On the Mapping Problem. *IEEE Trans. Comput.* C-30, 3 (March 1981), 207–214. <https://doi.org/10.1109/TC.1981.1675756>
- [2] Michael Booth, Steven P. Reinhardt, and Aidan Roy. 2017. *Partitioning Optimization Problems for Hybrid Classical/Quantum Execution*. Technical Report.
- [3] Rainer Burkard, Mauro Dell’Amico, and Silvano Martello. 2009. *Assignment Problems*. Society for Industrial and Applied Mathematics, PA, USA.
- [4] C. P. Chan, J. D. Bachan, J. P. Kenny, J. J. Wilke, V. E. Beckner, A. S. Almgren, and J. B. Bell. 2016. Topology-Aware Performance Optimization and Modeling of Adaptive Mesh Refinement Codes for Exascale. In *2016 First International Workshop on Communication Optimizations in HPC (COMHPC)*. 17–28.
- [5] Vasil S. Denchev, Sergio Boixo, Sergei V. Isakov, Nan Ding, Ryan Babbush, Vadim Smelyanskiy, John Martinis, and Hartmut Neven. 2016. What is the Computational Value of Finite-Range Tunneling? *Phys. Rev. X* 6 (Aug 2016), 031015. Issue 3. <https://link.aps.org/doi/10.1103/PhysRevX.6.031015>
- [6] NetworkX developers. [n. d.]. NetworkX. Software for complex networks. <https://networkx.github.io>.
- [7] Fujitsu. [n. d.]. Digital Annealer. <http://www.fujitsu.com/global/digitalannealer/>.
- [8] Fred Glover and Manuel Laguna. 1997. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA.
- [9] Torsten Hoefler and Marc Snir. 2011. Generic Topology Mapping Strategies for Large-scale Parallel Architectures. In *Proceedings of the International Conference on Supercomputing (ICS '11)*. ACM, New York, NY, USA, 75–84. <http://doi.acm.org/10.1145/1995896.1995909>
- [10] F. K. Hwang. 1987. The Hamiltonian Property of Linear Functions. *Oper. Res. Lett.* 6, 3 (July 1987), 125–127. [https://doi.org/10.1016/0167-6377\(87\)90024-1](https://doi.org/10.1016/0167-6377(87)90024-1)
- [11] D-Wave Systems Inc. [n. d.]. D-Wave. The quantum computing company. <https://www.dwavesys.com>.
- [12] D-Wave Systems Inc. 2015. *The D-Wave 2X Quantum Computer. Technology Overview*. Technical Report.
- [13] Tadashi Kadowaki and Hidetoshi Nishimori. 1998. Quantum annealing in the transverse Ising model. *Phys. Rev. E* (1998), 5355.
- [14] J. King, S. Yarkoni, J. Raymond, I. Ozfidan, A. D. King, M. M. Nevisi, J. P. Hilton, and C. C. McGeoch. 2017. Quantum Annealing amid Local Ruggedness and Global Frustration. *ArXiv e-prints* (Jan. 2017). arXiv:1701.04579
- [15] Soo-Young Lee and J. K. Aggarwal. 1987. A Mapping Strategy for Parallel Processing. *IEEE Trans. Comput.* C-36, 4 (April 1987), 433–442.
- [16] S. Mandrà, Z. Zhu, W. Wang, A. Perdomo-Ortiz, and H. G. Katzgraber. 2016. Strengths and weaknesses of weak-strong cluster problems: A detailed overview of state-of-the-art classical heuristics versus quantum approaches. *ArXiv e-prints* 94, 2, Article 022337 (Aug. 2016), 14 pages. arXiv:1604.01746
- [17] Michael A. Nielsen and Isaac L. Chuang. 2011. *Quantum Computation and Quantum Information: 10th Anniversary Edition* (10th ed.). Cambridge University Press, New York, NY, USA.
- [18] J. M. Orduna, F. Silla, and J. Duato. 2001. A new task mapping technique for communication-aware scheduling strategies. In *Proceedings International Conference on Parallel Processing Workshops*. 349–354.
- [19] S. Pakin. 2016. A quantum macro assembler. In *2016 IEEE High Performance Extreme Computing Conference (HPEC)*. 1–8.
- [20] Scott Pakin and Steven P. Reinhardt. 2018. A Survey of Programming Tools for D-Wave Quantum-Annealing Processors. In *High Performance Computing*, Rio Yokota, Michèle Weiland, David Keyes, and Carsten Trinitis (Eds.). Springer International Publishing, Cham, 103–122.
- [21] R. Salimi, H. Motameni, and H. Omranpour. 2012. Task scheduling with Load balancing for computational grid using NSGA II with fuzzy mutation. In *2012 2nd IEEE International Conference on Parallel, Distributed and Grid Computing*.
- [22] Satu Elisa Schaeffer. 2007. Survey: Graph Clustering. *Comput. Sci. Rev.* (2007), 38.
- [23] K. Taura and A. Chien. 2000. A heuristic algorithm for mapping communicating tasks on heterogeneous resources. In *Proceedings 9th Heterogeneous Computing Workshop (HCW 2000) (Cat. No.PR00556)*. 102–115.
- [24] Tony T. Tran, Minh Do, Eleanor G. Rieffel, Jeremy Frank, Zhihui Wang, Bryan O’Gorman, Davide Venturelli, and J. Christopher Beck. 2016. A Hybrid Quantum-Classical Approach to Solving Scheduling Problems. In *Ninth Annual Symposium on Combinatorial Search*.
- [25] Z. Wang, W. Liu, J. Xu, B. Li, R. Iyer, R. Illikkal, X. Wu, W. H. Mow, and W. Ye. 2014. A Case Study on the Communication and Computation Behaviors of Real Applications in NoC-Based MPSoCs. In *2014 IEEE Computer Society Annual Symposium on VLSI*. 480–485.
- [26] S Wayne Bollinger and Scott Midkiff. 1988. Processor and Link Assignment in Multicomputers Using Simulated Annealing.. In *ICPP*, Vol. 1. 1–7.
- [27] Yoshihisa Yamamoto, Kazuyuki Aihara, Timothee Leleu, Ken-ichi Kawarabayashi, Satoshi Kako, Martin Fejer, Kyo Inoue, and Hiroki Takesue. 2017. Coherent Ising machines—optical neural networks operating at the quantum limit. *npj Quantum Information* 3, 1 (2017), 49. <https://doi.org/10.1038/s41534-017-0048-9>
- [28] Kenneth M. Zick, Omar Shehab, and Matthew French. 2015. Experimental quantum annealing: case study involving the graph isomorphism problem. *Scientific Reports* 5 (08 06 2015), 11168 EP -. <http://dx.doi.org/10.1038/srep11168>