Benchmarking Quantum(-Inspired) Annealing Hardware on Practical Use Cases

Tian Huang, Jun Xu, Tao Luo, Xiaozhe Gu, Rick Goh, and Weng-Fai Wong, Senior Member, IEEE

Abstract—Quantum(-inspired) annealers show promise in solving combinatorial optimisation problems in practice. There has been extensive researches demonstrating the utility of D-Wave quantum annealer and quantum-inspired annealer, i.e., Fujitsu Digital Annealer on various applications, but few works are comparing these platforms. In this paper, we benchmark quantum(-inspired) annealers with three combinatorial optimisation problems ranging from generic scientific problems to complex problems in practical use. In the case where the problem size goes beyond the capacity of a quantum(-inspired) computer, we evaluate them in the context of decomposition. Experiments suggest that both annealers are effective on problems with small size and simple settings, but lose their utility when facing problems in practical size and settings. Decomposition methods extend the scalability of annealers, but they are still far away from practical use. Based on the experiments and comparison, we discuss the advantages and limitations of quantum(-inspired) annealers, as well as the research directions that may improve the utility and scalability of these emerging computing technologies.

Index Terms—Benchmark, combinatorial optimisation, digital annealer, quantum annealer

1 INTRODUCTION

Quantum annealing is a generic method for solving combinatorial optimisation problems by exploiting the quantum effects. The motivation for solving the problem with quantum annealer is speed. Many combinatorial optimisation problems, despite their simple problem settings, are computationally difficult. The problem-solving process utilises quantum fluctuation-based computation instead of classical computation. A combinatorial optimisation problem can be modelled as Quadratic Unconstrained Binary Optimisation (QUBO) form [1], which corresponds naturally to the transverse Ising model and benefits from the speed up by quantum annealing [2]. Some problems, e.g., max-cut problem [3] and max-sat problem [4], are especially popular among the quantum computing research community because of the simplicity of the problem settings. Applications with more advanced problem settings have attracted more attention recently. Examples are portfolio optimisation in finance [5], traffic flow management [6], warehouse management problem [7].

Although there are plenty of works demonstrating the capability of quantum annealing, the utility of quantum annealer is restricted by today’s manufacturing technology. For example, D-Wave Quantum Annealer (QA) has a limited number of qubits. Furthermore, the architecture of a Quantum Processing Unit (QPU) puts restrictions on the connectivity of the graph that represents the problem. On the other hand, Fujitsu Digital Annealer (DA) is a quantum-inspired CMOS-based ASIC, which simulates the annealing process efficiently. It enjoys speedup over general-purpose classical computers and has fewer restrictions than QA does.

One may ask which one, QA or DA, is better for combinatorial optimisation problems. As far as we know there is no comprehensive benchmark between them. The answer to the question depends on the problem settings. In this paper, we benchmark QA and DA with different combinatorial optimisation problems, ranging from simple to complex problem settings in practical use. Through analysis on experiments, we discuss the advantages and disadvantages of quantum(-inspired) annealers under different problems settings.

Due to the limitations of today’s manufacturing techniques, the capacity of a quantum(-inspired) annealer is often far away from the problem scale in practical use. Decomposition techniques are essential when the size of a problem goes beyond the capacity of a computer. This paper also benchmarks the quantum(-inspired) annealers in the context of decomposition. We use warehouse assignment problem as a case study and propose a new decomposition heuristic and compare it with various other methods. In this paper, We have the following contributions:

- To our best knowledge, this is the first work to systematically compare the performance between quantum annealer and digital annealer. We include combinatorial optimisation problems in different levels of complexity and quantitatively evaluate how these annealers respond to different problem settings.
We use the warehouse assignment problem as a case study and propose a new heuristic that enables quantum-inspired annealers to solve the warehouse management optimisation of size linear to the number of qubits, provided a certain block-structural property is satisfied.

We identify the advantages and missing pieces for QA and DA through a comprehensive analysis of the experiments.

We have the following observations from experiments: First, both QA and DA are very fast and efficient if the problem has a small scale and sparse connectivity between decision variables. Although QA and DA have different working principles, both of the performance, in terms of quality of solutions, decrease when the size of the problem scales up. Introducing constraints into the problem will add difficulty for QA in finding optimal or even feasible solutions. Dense connectivity reduces the utility of QA. DA is more robust to these challenges. DA has a better scaling factor compared with CPU-based simulated annealing. It is effective on constrained and densely connected problems, which are more common in real-world applications.

Second, our new decomposition heuristic for warehouse management is based on the assumption of a block-structural property. Such property is frequently observed in many practical settings including warehouse allocation. Theoretical proof provides a boundary for the performance of the problem-solving process. Experiments suggest that our new decomposition heuristic produces more promising solutions with less memory consumption than QBSolv does and enjoys speedup over simulated annealing on generic purpose classical CPUs. However, experiments show that both hybrid methods lose their utility as the problem size continues to scale up.

Through the experiments, we conclude the following directions to improving the performance of quantum (-inspired) annealers: 1. Error and noise mitigation; 2. Constraints simplification and post-readout processing; 3. Smart encoding; 4. Smart decomposition.

2 RELATED WORK

2.1 Annealing-Based Computers

Quantum Annealing (QA) is a method that minimises the energy of an objective function. It takes advantage of quantum mechanics. Quantum bit or qubit is the basic unit in a quantum computing system, which resembles the binary bit in classical digital circuits. A coupler in quantum annealer bonds two qubits together and allows quantum mechanics to work its magic. Qubits and couplers in a quantum annealer can be seen as the nodes and edges in a graph. To use a quantum annealer, one has to convert a problem into a Quadratic Unconstrained Binary Optimisation (QUBO) [1] form and map the QUBO problem onto the quantum annealer.

D-Wave is a world-leading company that designs and builds quantum annealers. The Quantum Processing Unit (QPU) in the D-Wave’s quantum annealing system is a lattice of interconnected qubits. Each qubit is made of a superconducting loop. D-Wave released the D-Wave 2000Q system in 2017, the QPU in which employs a Chimera graph architecture [8], equipped with 2048 qubits and 6016 couplers. In 2019, D-Wave released the Advantage system, the QPU of which employs a Pegasus [9] graph architecture, which has 5640 qubits and 40,484 couplers.

The Fujitsu Digital Annealer (DA) [10] is a hardware implementation of an enhanced variant of the Simulated Annealing (SA) algorithm. DA employs Metropolis-Hastings update with Parallel Tempering. In each cycle, it adjusts the temperature, proposes bit flips, and accepts/rejects the proposals according to the temperature and periodically swaps configurations between systems. Apart from the sophisticated techniques in proposing and selecting updates, the competence of DA also comes from the efficient hardware implementation that exploits the parallelism in the algorithm.

2.2 Benchmark for Quantum(-Inspired) Annealing

There are many works demonstrating the utility of quantum and digital annealers in various applications [3], [4], [5], [6], [7], [11], [12], [13]. Bass et al. [14] did a performance benchmark for four hybrid methods, in which the largest problem requires 10,000 binary decision variables. Vert et. al. [8] investigated the relationship between the quality of solutions and the connectivity of decision variables for hybrid methods. Ohzeki et al. [15] compares the performance of quantum annealer and digital annealer on automated guided vehicles (AGV) problems. Albash et. al. [16] compares the scalability of quantum and classical annealing. Seker et. al. [17] systematically investigate the performance of DA on various problems. Kowalsky et.al. [18] perform a scaling and prefactor analysis of the performance of four quantum(-inspired) optimizers on 3-regular 3-XORSAT problems.

None of these works provides a comprehensive comparison between a quantum annealer and a digital annealer. In this paper, we evaluate the two annealers on three representative combinatorial optimisation problems and compare their performance qualitatively and quantitatively. In the context of decomposition, the largest problem instance in this paper involves 8100^2 = 65.61 \times 10^4 binary decision variables, which is close to the size of the application in practical use.

2.3 QUBO Decomposition

In a large neighbourhood local search, a random feasible solution is usually taken as the starting point and a small subset of variables solvable on hardware are chosen in each iteration. The subproblem generated by this subset is solved while variables outside of the subset are conditioned. This yields an exploration of a very large neighbourhood, whose result can be optionally plugged into a complementary software local search that gets to a local minimum. Overall, this approach has two complementary steps: the perturbation/jump step that find a point with a good neighbourhood to work with, and the local search step for finding the local optimum within such a neighbourough. QBSolv is the canonical D-Wave implementation [19] that uses subsets of variables for perturbation. A comprehensive evaluation [20] shows QBSolv outperforms many other decomposition methods. There is another implementation by Oracle [21] which inverts the way by solving random subsets of variables for local search and updating randomly chosen variables for perturbation. The approach described in this paper
falls into this category but is a special case without perturbation. A neighbourhood is a-priori determined and subsets of variables are chosen heuristically on a higher abstraction level instead of at random.

Much of the quantum annealer application work cited in Section 1 is experimental in scale and does not deal directly with Quadratic Assignment Problem (QAP). QAP asks for optimal assignment of \( n \) facilities to \( n \) locations that minimise the logistic efforts between them. There is only one instance of QAP being solved on a quantum device with application in flight gate assignment [22]. Just like what we mentioned in Section 1, the QAP instance in [22] is too big for DA. The scaling method they used is to randomly divide a graph into disconnected components if it is too large, and solve each component separately. This approach, while scalable, is a brute-force one based on randomness, and does not give significant insight as to how a QAP can be meaningfully decomposed.

3 On Direct Problems

In this section, we demonstrate the advantages and limitations of quantum(-inspired) computers by applying them to three representative combinatorial optimisation problems. The demonstration of problems is arranged progressively, such that a latter problem setting poses more challenges, which further weaken the utility of a quantum(-inspired) annealer. We try to answer the following three questions empirically:

- What problem size can a quantum(-inspired) annealer handle?
- How much performance improvement can a quantum annealer achieve compared to a classical one?
- How is the quality/optimality of the solutions found by a quantum annealer compared to those of a classical one?

We include the D-Wave QPU of Chimera and Pegasus architecture, and Fujitsu Digital Annealer (DA) in the evaluation. We also include Simulated Annealing (SA) by D-Wave, which is an open-source CPU-based implementation of the Metropolis-Hastings algorithm. We include QBSolv from D-Wave, which is a heuristic hybrid solver that incorporates classical computer and quantum annealer. We also include Gurobi, which represents the SOTA commercial optimisation solver. We use Gurobi to find the global optimima of the synthetic datasets in this paper.

The annealing process is crucial to the performance of annealing-based solvers. We try out different hyper-parameters for the annealing process. More specifically, we are varying annealing time, or \( T_{\text{total}} \), in D-Wave QA, number of iterations, or \#iter, in Fujitsu DA and number of sweeps, or \#sweep, in SA. We only include the results with the best hyper-parameters in the comparison in the main text. Our definition of “best” priorities energy functions and take timing into consideration as well. Readers can find the details of the experimental settings in Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2022.3219257 and complete results in Appendix C, D and E, available in the online supplemental material.

3.1 Max-Cut

The max-cut problem is a well-known combinatorial optimisation problem that aims to find a partition of a node set into two parts, maximising the sum of weights over all edges across the two node subsets of a graph. Such a partition is called a maximum cut. Consider an undirected graph \( G = (V, E) \), where \( |V| = n \) with edge weights \( w_{i,j} > 0 \), \( w_{i,j} = w_{j,i} \) for \( (i, j) \in E \). We partition \( V \) into two subsets. The cost function to be optimized is the sum of the weights of edges between two subsets of \( V \). The definition of the cost function is given by

\[
\sum_{(i,j) \in E} w_{i,j}x_i(1-x_j),
\]

where \( x_i \) and \( x_j \) are binary decision variables. \( x_i \) is set to 0 or 1 depending on whether the node \( i \) is in the first or second subset, respectively. The goal is to find a partition that maximises the cost. One solution would be in the form of \( x_0, \ldots, x_{n-1} \). An annealer must have at least \( n \) qubits to handle a max-cut problem with \( n \) nodes.

3.1.1 Pegasus-Like Max-Cut Problems

We formulate ten max-cut problems of different sizes, each of which is a sub-graph of the Pegasus graph architecture. The number of nodes of a graph \( |V| \) ranges from 543 to 5430.

The reason for generating Pegasus-like graphs is two folds. First, we want to understand the performance of a quantum annealer when we make full use of the resources on the QPU. The largest problem that can be handled by a QPU is related to the match between the problem and QPU’s architecture. If the graph of a problem is the same as or is a subset of the architecture, then the problem can be easily mapped onto the QPU. Otherwise, extra qubits are needed to map the problem. Using Pegasus-like graphs minimises the extra cost and maximises the utilisation of resources on the QPU.

Second, we want to minimise the uncertainties in measuring the performance of the QPU. To handle a problem that does not match with the architecture very well, an extra step called minor embedding [23] is needed to map the problem onto the QPU. Minor embedding itself is an NP-hard problem and usually introduces extra qubit cost and extra noises to the system, which weaken the performance of the QPU. Pegasus-like problems can be directly mapped onto a Pegasus architecture without a resort to minor embedding, and therefore minimises the uncertainties in the measurement.

Fig. 1a shows the timing of the solvers on the Pegasus-like max-cut problems. Pegasus QPU spends about 200 ms on all these problems. The total execution time of the QPU can be decomposed into programming time and sampling time. The sampling time is fixed because the QPU is based on the same annealing process. The programming time increases along with the problem size. Pegasus is generally
2-3 orders of magnitude faster than the classical annealing-based solvers. The difference in timing mostly comes from the fundamental difference in the working principles. The solving time of classical solvers depend on the size of a problem. Larger problem usually requires longer running time. The solving time of Gurobi increases sub-exponentially along with the size of the problem. Because we set the Gurobi program to terminate when its running exceeds ten hours, the second half of the Gurobi curve clips.

Fig. 1b shows the objective energy, normalised to that of Gurobi. Two statistics, mean energy (dotted lines) and best energy (solid lines), are included. For hyper-parameters, we use $T_{\text{neal}} = 2000\mu s$ for D-Wave annealer, $\#\text{iter} = 10^6$ for DA, and $\#\text{sweep} = 10^4$ for SA.

Fig. 2 shows the objective energy, normalised to that of Gurobi. To describe the statistics of the results, we use solid curves to represent the energy of the best solution, and use the dotted curves to represent the mean energy of the solution. Since max-cut is a maximisation problem, higher energy is better. The energy achieved by classical annealers and quantum annealer is not closely related with problem size. This experiment demonstrates the scalability of classical and quantum annealers. Pegasus outperforms all classical annealers on most of the problem instances by a margin of 1% – 3% energy. DA performs better on the problem instances of size over 4000, which have higher connectivity than what smaller instances have.

We also have experiments on Chimera-like max-cut problems. Please find the details in the Appendix C.2, available in the online supplemental material.

2. The mean-best plot setting is a replacement of an error-bar setting. The latter one introduces overlapping issues and is hard to read. Readers can find per-solver error-bar plot setting for a range of hyper-parameters in Appendix C, D, and E, available in the online supplemental material.

3. This is the largest complete graph Pegasus accepts. For DA, the up-limit is 8192. To compare Pegasus and DA, we have to use 145.

3.1.2 Connectivity-Variied Max-Cut Problems
The Pegasus-like graph in the previous section is sparsely connected graphs and are not in favour of DA. According to Fujitsu’s official documentation [10], DA is designed for solving optimisation problems with densely connected graphs. In this section, we examine how connectivity affects the performance of an annealer. We make 32 randomly generated max-cut graphs, each of which has 145 nodes. The average degree of these graphs ranges from 1 to 140. For the Pegasus QPU, there will be auxiliary qubits and connections involved in the final embedding, since the connectivity-varying problems are mostly not a sub-graph of the Pegasus graph architecture.

Fig. 2a shows the time consumption of all solvers. Problems with degree over ten are challenging such that Gurobi cannot quickly traverse the solution spaces, so we set a timeout limit of ten hours. On the other hand, there are only 145 nodes in these graphs, which is “trivial” for classical annealers. DA, SA and QBSolv finish searching within one second, which is very close to the timing of Pegasus.

Fig. 2b shows the objective energy of the solutions. All solvers, except Pegasus, can achieve 100% energy on almost all problem instances. The mean energy of DA improves as the connectivity increases. We have similar observations in Section 3.1.1, where the energy of DA improves as problem size increases. This suggest that DA performs better on densely connected problems. In comparison, the energy of Pegasus decreases when averaged degree is over 12. The mean energy could drop down to as low as 85%, which
corresponds to averaged degree of 16. This suggests the Pegasus performs better on sparsely connected problems.

The problem setting of max-cut is straightforward. Many other problems have constraints that cannot be directly represented in existing annealers. An annealer may find a solution that violate the constraints. We shall see how the situation changes for such problem settings.

### 3.2 Minimum Vertex Cover

Given an undirected graph with a set of nodes $V$ and edges $E$, a vertex cover of a graph is a set of nodes that includes at least one endpoint of every edge of the graph. The Minimum Vertex Cover (MVC) problem is an optimisation problem that finds the smallest vertex cover of a given graph. MVC problem can be formulated as Eq. (2)

$$
\text{Minimize } \sum_{v \in V} x_v \quad \text{subject to } x_i + x_j \geq 1, \forall (i, j) \in E, \quad (2)
$$

$x_i = 1$ indicates node $i$ is in the cover and $x_j = 0$ otherwise. The objective function minimises the number of nodes in the vertex cover, whereas the constraints ensure that at least one of the endpoints of each edge $(i, j)$ will be in the cover. In a quantum-inspired (annealer, there is no corresponding mechanism to ensure the satisfaction of the constraints. A workaround is to lift the energy of configurations that violates the constraints and such that it becomes a less favourable solutions. This method is widely known as “penalty”. The penalty of MVC can be formulated in the form of $a \cdot (1 - x_i - x_j + x_i x_j)$, where $a$ represents a positive scalar penalty. A QUBO form for MVC is given by

$$
\text{Minimize } \sum_{v \in V} x_v + a \sum_{(i, j) \in E} (1 - x_i - x_j + x_i x_j). \quad (3)
$$

The penalty term does not introduce new connections to the graph. The QUBO form has the same topology as that of the original problem. The choice of penalty weight $a$ is application-specific. For Eq. (3), adding a node to a minimum vertex cover will increase the objective energy by one. Removing any node from a minimum vertex cover will increase the objective energy by $a - 1$. [1] suggest any $a > 1$, for example, $a = 2$ would ensure that a solver can find solutions that satisfy the constraints of a MVC problem.

#### 3.2.1 Connectivity-Varied MVC Problems

We reuse the connectivity-varied graphs in Section 3.1.2 for constructing MVC problems. We discover that some solvers have difficulty in finding feasible solutions with penalty weight $a = 2$. We set $a = \max_{v \in V} (i)$, which represents the highest degree in a graph. With this setting, we can understand the comparison between the annealers more clearly.

Fig. 3a shows the timing of the solver on the connectivity-varied MVC problems. Since we are using $T_{\text{neal}} = 2000\mu$s for Pegasus, it loses its leading position in terms of timing. This figure is similar to Fig. 2a, except the timing of Gurobi stops increasing when the averaged degree is above 10. All solutions obtained by Gurobi are global optimal solutions.

Fig. 3b shows the objective energy of solutions, normalised to that of Gurobi. Since MVC is a minimisation problem, lower energy is better. The best energy curve of DA is always at 100%, which means it finds optimal solutions for all problems. It is overlapped with Gurobi. One can spot the markers of DA and Gurobi by zooming in. Pegasus find the optimal solution only when the average degree is equal to one. Then there is more than 20% performance drop as the degree increases. The Pegasus curve does not continue beyond a degree of 11 as it fails to find any feasible solution.

In annealing-based methods, we sample multiple times for a problem instance. If a problem has constraints, some samples of the QUBO form may not be feasible, i.e., does not satisfy with constraints. We introduce a metric called probability of feasibility, or $P_f$

$$
P_f = \frac{\text{Feasible samples}}{\text{Total samples}} \times 100%. \quad (4)
$$

Fig. 4a shows the relation between averaged degree and $P_f$. The SA curve is always 100%, which provides a sanity check of the choice of the penalty coefficient $a$. The Pegasus curve drops to zero when the degree goes beyond 11. This is where the Pegasus curve stops in Fig. 3b. The Pegasus curve suggests it is struggling to find feasible solutions as the average degree increases. In fact, for the problem with 140 degrees, we have fine-tuned $a$, or even remove the objective and solve the constraints solely, and still cannot find any feasible solution. DA in Fig. 4a achieves about 80% $P_f$. With higher $\#\text{iter}$, e.g., $10^6$, it achieves 100% all the time. Gurobi always find feasible solutions because the constraints can be explicitly implemented in Gurobi and directly regulate the searching process.

We further investigate the constraint violation in a solution. This is for us to understand how easy it is to fix the
violation and salvage promising solutions from broken ones. For MVC, the number of constraints is equal to the number of edges in a graph, according to Eq. (2). If two decision variables on two sides of an edge are all zeros, we count it a violation. We divide the number of violations in a solution by the number of constraints, and name the result as the percentage of constraint violation. We use pseudo energy to measure the quality of infeasible solutions, which is the original objective energy plus the non-zero penalty terms.

We examine the problem of degree equals to 10, because this is one of the most challenging instances, according to the time consumption of Gurobi. The relation between energy and constraint violation is shown in Fig. 4b. Each marker represents a solution. The left bottom corner, i.e., the coordinate (0.0, 1.0), is the global optima. The solutions of QBSolv and SA reside on the line of violation=0.0%, which means these solutions are feasible. On the other hand, many solutions from Pegasus distribute across 0%-1%, which are infeasible solutions. The distribution of the infeasible solutions of DA is similar to a Pareto frontier, in which if you emphasize more on fewer violations, you get higher energy. In some cases, one can see a similar frontier in quantum annealer’s constraint violations, which is available in Appendix D and E, available in the online supplemental material. Fixing broken constraints is another broad topic, which falls out side the scope of this paper. If fixation is not an option, one can also use these infeasible solutions as initialisation of another search.

3.2.2 DIMACS 10th Challenge

We further evaluate the annealers on real and random datasets extracted from the 10th DIMACS Challenge [24]. Eleven graphs are included in this benchmark. The number of nodes ranges from 34 to 22963. Please find the detailed of dataset in Appendix D3, available in the online supplemental material.

In Fig. 5a, the timing of the solvers remains a similar trend compared to previous experiments. The Pegasus curve stops when the size of the problem is over 1024 because D-Wave’s minor embedding heuristic failed to map larger problems onto Pegasus. DA cannot handle the problem size over 8192.

According to Fig. 5b, DA outperforms Pegasus in terms of the quality of solutions, as well as problem size. If we check with Fig. 6a, we know that Pegasus is struggling to find feasible solutions, when the problem size is over 100. This matches with the results in previous section. In comparison, the solutions found by DA are mostly feasible.

Fig. 6b shows the relation between normalised energy and constraint violation on the famous “delaunay_n10” problem. We choose it because this is the largest that Pegasus can handle in this dataset. Most of the solvers have similar distributions of solutions, compared with Fig. 4b. For Pegasus, the distribution is not similar to a Pareto frontier. The normalised energy for Pegasus is around 1.18 and is independent of the constraint violation. We see this as a sign that the noises and errors overwhelm in the quantum annealer, such that the solutions are no longer related to the original problem.

We use the benchmark from DIMACS 10th Challenge in 2012, because we believe it closely follows the trend of
requirements nowadays. However, most of the previous works on heuristic MVC methods are based on the benchmark from DIMACS 2nd Challenge in 1992. We evaluate benchmark from DIMACS 2nd as well and include the results in Appendix D.4, available in the online supplemental material. We also have experiments on a set of Pegasus-like MVC problems in Appendix D.1, available in the online supplemental material.

3.3 Quadratic Assignment Problem (QAP)

The Quadratic Assignment Problem (QAP) is a well-known combinatorial optimisation problem with a wide range of applications, such as backboard wiring, statistical analysis, placement of electronic components, etc.

QAP can be visualised as the problem of assigning \( n \) facilities to \( n \) locations. Between any two facilities, there are products transported back and forth, and the amount transported is called the \textit{flow}. Between two locations there is some distance. \textit{Traffic} is defined as the product of flow and distance. The problem is to assign facilities to locations such that the sum of traffic is minimised. The aim is to find an assignment of the facilities to the locations such that the sum of all products of flows and their corresponding distances is minimised, subject to the constraints that each facility is assigned to exactly one location and each location contains exactly one facility.

Formally, we assume \( n \) is the number of facilities which is also the number of locations. \( F = (f_{ij}) \) is the flow matrix for facilities. Each entry represents the flow between facilities \( i \) and \( j \). \( D = (d_{ij}) \) is the distance matrix. Each entry represents the distance between location \( i \) and \( j \). \( X = (x_{ij}) \) is the binary decision matrix. Each entry represents whether facility \( i \) is assigned to location \( j \). Note that the dimension of \( X \) is \( n \times n \), which implies that there are \( O(n^2) \) binary decision variables. If the flow is given as the matrix \( (f_{ij}) \) and distance is given as the matrix \( (d_{ij}) \), then the objective function to minimise is

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} f_{ij} d_{ik} x_{ij} x_{jk}
\]

subject to \( \sum_{k=1}^{n} x_{ik} = 1 \quad \forall 1 \leq i \leq n \) \hfill (5)

and \( \sum_{i=1}^{n} x_{ik} = 1 \quad \forall 1 \leq k \leq n \) \hfill (6)

where \( x_{ij} \) is the binary decision variable representing whether facility \( i \) goes to location \( k \). The constraints ensure that one facility is assigned to a location exactly once.

To enable the annealing of a QAP, we have to subsume constraints Eqs. (6) and (7) as a part of the objective function Eq. (5). Conventionally, we can encode the constraints as penalty terms which augment the objective function.

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} f_{ij} d_{ik} x_{ij} x_{jk}
+ \alpha \left( \sum_{k=1}^{n} x_{ik} - 1 \right)^2 + \beta \left( \sum_{i=1}^{n} x_{ik} - 1 \right)^2
\]

Eq. (8) is a standard QUBO form of QAP. \( \alpha \) is a penalty weight, which connects the original objective function Eq. (5) and the constraints Eqs. (6) and (7). It can be shown that for QUBOs, the optimal solution to the augmented objective function also minimises the original objective function.

3.3.1 Tinyqap

A QAP with \( n \) facilities and \( n \) locations have \( n^2 \) variables, which often yields large models in practical settings. Compared to the max-cut and MVC problems, QAP is more demanding on the number of qubits. All problems in QAPLIB [25] are beyond the capability of a quantum annealer.

We generate a set of tiny QAP problems to facilitate the evaluation of quantum annealers. The problem size ranges from 3 to 12. We set penalty weight \( \alpha = n \times \max(|C|) \), where \( C \) represents a list of the coefficients of all linear and quadratic terms in Eq. (5). This setting follows the suggestion in [1] about the “Scalar Penalty \( P_s \)” with a little more emphasis on the feasibility. The settings are applied to all experiments in Section 3.3.

In Fig. 7a, Chimera and Pegasus are not the fastest. The Chimera curve stops at 8, which corresponds to 64 decision variables. The largest complete graph that Chimera accepts is 65. QAP of \( n = 12 \) is the up-limit of a Pegasus QPU can handle. DA is not designed for such trivial optimisation problems. We can see that the timing of DA does not scales up as the problem size increases.

As shown in Fig. 7b, Chimera only finds feasible solutions when \( n \leq 4 \). From Fig. 8a We know that the \( P_f \) of Chimera quickly drops to 0% when \( n > 4 \). The Pegasus QPU
managed to find feasible solutions for \( n = 5 \), but its \( P_f \) is very close to that of Chimera. This suggests Pegasus architecture does not have a clear advantage over the Chimera architecture. Please note that we are using quite large \( T_{\text{anneal}} \) for Chimera and Pegasus, which generally leads to better feasibility. For Chimera, increasing \( T_{\text{anneal}} \) by a factor of 10 does not give any improvement in terms of \( P_f \), so we just demonstrate the results of \( T_{\text{anneal}} = 200 \) ms. DA can find very promising solutions, which are all feasible and very close to the global optima.

We also investigate the constraint violation of the solutions to “tiny-08,” which allows us to compare Chimera and Pegasus. The constraints in QAP are one-to-one mapping constraints. For example, in the tiny-03 problem, we have 6 constraints, i.e., 3 constraints to ensure each facility is only assigned to one location, and another 3 constraints to ensure each location only accommodates one facility. From Fig. 8b we see the solutions of QBSolv and DA reside at coordinate (0.0, 1.0), which means these solutions are all feasible and very close to the global optima.

In comparison, the two quantum annealers are suffering from many constraint violations. From Fig. 8b we see the energy distribution of Chimera and Pegasus is roughly proportional to the number of violations. This is partly because we put more emphasis on the feasibility of a solution and put high penalty in the objective energy. With the same penalty settings, the annealing-based classical solvers find feasible and competitive solutions most times, but neither of the two quantum annealers can find any feasible solutions. We believe this is partly due to the misimplementation of the problems on the quantum annealers because of the analog control errors in QPUs. [26]. In this experiment the solutions of Chimera is closer to the optimal solution compared with Pegasus. However we cannot assert Chimera architecture is better than Pegasus, because we only have access to one QPU instance of each architecture and cannot rule out the impact of individual differences in the performance evaluation.

TinyQAP is too small for DA. We are going to push towards the limit of DA in the next experiment.

### 3.3.2 QAPLIB

We choose the Tai dataset from QAPLIB [25] because the size fits well with the purpose of our evaluation. We only use the instances Taixxa, which are uniformly generated. We do not include quantum annealers in this experiment because most of the problems are too large for them.

According to Fig. 9a DA has its advantage over SA when the problem size is over 30. It also outperforms SA and QBSolv in terms of objective energy in Fig. 9b. The hybrid method QBSolv find better solutions than SA, but not as good as DA. We also observe that both SA and QBSolv have difficulty in finding feasible solutions, according to Fig. 10a.

Fig. 10b is the constraint violation of solutions to “tai80a”. We choose this problem because it is the largest one in the dataset that can be handled by DA. All solutions of DA feasible. For SA and QBSolv, although none of their solutions is feasible, there is only about 3-15% of constraint violation, which means the violation fixation task is not very challenging.
Warehouse management is a representative application of QAP. A warehouse usually has items and locations that are much larger than the problem size in previous QAP datasets. We aim to develop a scalable method and facilitate Warehouse management problems with the power of quantum(-inspired) computers.

4 THE WAREHOUSE ASSIGNMENT PROBLEM

In warehouse management, order picking is a labour-intensive and costly activity. [28] Consider the problem of assigning items to locations in a warehouse. How should the storage assignment be done, so that when orders come in and items are picked, the travel distance of the picker is at a minimum? For a company, the reduction of such distance translates to the reduction of labour costs. The objective is to find a decent assignment that minimises the travel distance for a given set of orders.

Consider a warehouse with a layout of Fig. 11a. It has three aisles and 30 locations in total. Each aisle can reach the two columns on its left and right. There is an Input/Output (I/O) point at the bottom left corner through which items are transported in and out of the warehouse.

A set of orders is shown in Fig. 11b. There are 10 orders and 10 storage keep units (SKUs). An SKU represents a unique type of item. The 10 SKUs are labelled 1 to 10. There are a total of 30 items, each of which belongs to a particular SKU.

To pick an order, the picker always starts from the I/O point. It may follow a different route to collect all items. Among various routing strategies, S-shaped (traversal) is the most common and simplest routing policy used in practice. [28] With this policy, the order picker begins by entering the aisle closest to the I/O point. If an aisle contains at least one item, the picker traverses the aisle over the entire length, otherwise, the picker skips that aisle. After picking the last item, the order picker takes the shortest route back to the I/O point. In this paper, we applied a modified S-shaped routing policy, in which a picker will not skip an aisle if there is no item in it. The modification provides a better match between the objective function of the warehouse assignment problem and QAP. An example of the S-shaped routing policy is illustrated in Fig. 11c. The picker collects the highlighted item in the first and last aisle by following an S-shaped route even if there is no item to pick in the second aisle.

It is easy to apply QAP to the setting of warehouse assignments. [29] To do so, simply replace “facilities” with “items”. The flow between items is defined to be the frequency at which their respective SKUs appear in the same order. This can be computed based on the incoming order set. With our revised S-shaped routing policy, the distance between two items is roughly proportional to the number of columns between them, since the picker would in principle traverse those columns in a zig-zag manner.

4.1 Heuristics: Block Structure

What inspires the application of QAP in warehouse assignment is the idea to assign items with high flow closer to each other. Intuitively, this would reduce the amount of traffic in picking orders because the picker is more likely to travel a shorter distance in between items appearing in the order.
The block structure of QAP occurs in the distance matrix \(d_{ij}\). Suppose \(V(H)\) is the nodes in a graph of the locations. There is a partition of \(V(H)\) into \(K\) subsets and a function that maps each location to its respective subset: \(c : V(H) \rightarrow K\), where \(K = 1, 2, \ldots, k\), such that

\[
d(x, y) = \begin{cases} 
\delta(x, y) & \text{if } c(x) = c(y) \\
M(x, y) & \text{if } c(x) \neq c(y) 
\end{cases}
\]

where \(\delta\) and \(M\) are both functions. The concrete functions would depend on \(d\), but with block structure \(\delta\) would “usually” return smaller values than \(M\). A special case is when \(\delta\) and \(M\) are constant with \(\delta < M\), and \((d_{ij})\) will be a “block matrix”.

Consider an example with the following warehouse of 8 locations, with each location labelled as in Fig. 12a. Assume that the vertical distance is 1 and the horizontal distance is 3. The distance matrix looks like the one in Fig. 12b (only the upper triangular part is shown).

Noticeably, the \(4 \times 4\) sub-matrix in the upper right corner, corresponding to the distance between the 1st and the 2nd aisles, is uniformly larger in number than the sub-matrices near the diagonal. In general, we could relabel the indices of locations such that the columns of a warehouse form intervals, and that when we move away from the diagonal, the matrix entries become larger because the columns are further away.

Since the above block-structure is observed, one simplification is that all locations within the same column have distance 1 (i.e., \(\delta \equiv 1\) in between, and all locations in different columns have a larger distance, say 8 (i.e., \(M \equiv 8\)). Then the block structure becomes more pronounced as shown in Fig. 12c. Please note that the diagonal of the distance matrix represents the distance between the entry and a certain location. The calculation of the diagonal does not follow the block structure. That’s why the diagonal is the same in Figs. 12b and 12c.

The simplification would normally be expected to flatten the energy landscape and remove tall and thick energy barriers. This benefits annealing-based computers, since energy barriers are notoriously challenging to both quantum and classical annealers. Quantum annealers use quantum mechanics to overcome energy barriers. For classical annealing, the parallel tempering method in DA and many other methods are proposed to overcome energy barriers. A flattened energy landscape is generally easier for annealing-based solvers to work with.

Similar to the way we can perform partitioning on locations, the item set can also be partitioned into \(K\) of equal size, where \(K \) is fixed to be \(|K|\). Intuitively, we want items to appear together in high frequency to be assigned closer to each other. Therefore, we maximise the sum of interaction frequencies within the subsets and then assign each subset to a subset in the partition of \(V(H)\), the set of locations. Note that \(K\) divides \(n\) is assumed, so that locations and items can be divided evenly.

### 4.2 Decomposition

Next, we describe the decomposition formally and provide proof of the theoretical boundary. To minimise the travelling distance, intuitively, we want to maximise the interaction frequencies of items within a block. This objective can be formally described as follows:

\[
\text{Maximise } \sum_{i,j=1}^{n} \sum_{k=1}^{K} f_{ij} x_{ik} x_{jk}.
\]

Eq. (10) denotes the sum of interaction frequencies among items within their respective subsets. \((x_{ij})\) is an \(n \times k\) decision matrix. \(x_{ij}\) denotes whether item \(i\) goes to subset
j. The objective comes with the following constraints

\[ \sum_{i=1}^{k} x_{il} = 1 \quad \forall i, 1 \leq i \leq n \]  
(11)

\[ \sum_{l=1}^{n} x_{il} \leq s \quad \forall l, 1 \leq l \leq k \]  
(12)

Eq. (11) means an item belongs to exactly one subset. Note that each subset can have at most \( s = \frac{n}{2} \) items. Therefore, Eq. (12) means each subset must not exceed its capacity \( s \).

Note that Eq. (10) is actually another QAP with a particular decision matrix (but with different constraints). Thus, it can be translated into a graph formulation. Suppose \( V(G) \) and \( E(G) \) are the nodes and edges in a graph of items. In the graph formulation, \( (x_{ij}) \) is equivalent to a function \( g: V(G) \rightarrow K \), which maps an element of \( V(G) \) into its subset, such that \( g(i) = a \Leftrightarrow x_{ia} = 1 \). This function is well-defined due to Eq. (11). Eq. (10) can be converted to the equivalent graph formulation as follows:

\[ \sum_{i,j=1}^{n} \sum_{l=1}^{k} f_{ij} x_{il} x_{jl} = \sum_{(i,j) \in E(G)} f(i,j), \]  
(13)

which, intuitively, is the sum of flows within subsets. This is the setup for the theorem below, which states that with a solution to Eq. (10), an optimal solution to the overall QAP Eq. (5) can be constructed. The proof is available from Appendix F.5, available in the online supplemental material.

**Theorem 1.** Let \( d(x, y) = \begin{cases} \delta & c(x) = c(y) \\ M & c(x) \neq c(y) \end{cases} \) for some positive constants \( \delta \) and \( M \). Then there exists a \( q_0 \) for which Eq. (5) is achieved, and \( q_0 \) can be constructed from a solution to Eq. (10) denoted by \( g_0 \).

This theorem assumes a strong condition that \( (d_{ij}) \) only has two distinct values, \( \delta \) for locations within an interval and \( M \) in between intervals. This is a simplification of the block structure and \( (d_{ij}) \), in reality, is usually more complex.

However, the concept of intervals can be generalised such that when \( \delta \) and \( M \) are non-constant, a function \( c \) can still be defined on \( V(H) \) such that the sum of distances within intervals is minimised. The motivation for such construct is that when a warehouse does not exhibit a clear-cut column structure, one can still think of an abstract “column” as a group of close locations. Note that this definition automatically specialises to the previous definition of \( c \) when \( \delta \) and \( M \) are constant. This intuition can be formally expressed as

\[ \text{Minimise} \quad \sum_{i,j=1}^{n} \sum_{l=1}^{k} d_{ij} x_{il} x_{jl} \]  
(14)

Subject to

\[ \sum_{i=1}^{k} x_{il} = 1 \quad i \in [1, n] \]  
(15)

and

\[ \sum_{i=1}^{n} x_{il} \leq s \quad l \in [1, k] \]  
(16)

\( (x_{ij}) \) is an \( n \times k \) decision matrix. \( x_{ij} \) denotes whether item \( i \) goes to location group \( j \). Note that this graph partitioning problem can be thought of as the dual to Eq. (10), with maximisation changed to minimisation. In effect, items that are frequently ordered together will be assigned to locations that are closer together.

Now both the set of locations and the set of items have been divided into subsets of equal size. Between subsets of locations, the distances are maximal, and between subsets of items, the interaction frequency is minimal. The final step is to produce a bijection from the set of subsets of items to the set of subsets of locations, and for each pair of subsets in the bijection (items, locs), there is a sub-QAP of size \( O(\sqrt{n}) \), and therefore a QUBO of size \( O(n) \), where \( n \) is the total number of items.

Note that this is subject to \( n \) being a perfect square; in practical situations where \( n \) is not a perfect square, compromise has to be made in either finding the nearest smaller perfect square and do the optimisation on the smaller set of items and locations only, or use integer divisors of \( n \) other than \( \sqrt{n} \), and correspondingly deal with sub-QAPs of sizes other than \( \sqrt{n} \). For example, if there are \( n = 3600 \) locations, it is possible to divide it into 40 groups of 90, where each sub-QAP will have \( n = 90 \).

The overall procedure runs the decomposition in Subsection 4.2 and then solves each individual sub-QAP using the exterior penalty method. Each sub-QAP utilises a conversion procedure to transform it into QUBO solvable by quantum (inspired) annealer. Finally, the procedure forms global solutions by aggregating solutions to sub-problems. The details of the overall procedure, the exterior penalty method and the conversion procedure are available in Appendix F.6 and F.7, available in the online supplemental material.

## 5 Experiments

The experiment has three objectives, namely 1) to evaluate the performance of the QAP decomposition heuristic in solving block-structural QAPs of various sizes, 2) to evaluate the effectiveness of the heuristic in minimising the warehouse picking distance, and 3) compare the performance of various computing hardware using different decomposition heuristics. In addition, we also included the heuristic algorithm “Tabu search” as a reference.

### 5.1 Comparison on QAP

We compare the performance of decomposition on QAP. Block-structural QAP instances WH-8, WH-90, WH-180, WH-270, WH-3600, and WH-8100 are synthetised using randomly generated order sets. The number corresponds to the size of a problem. The results obtained for each of the datasets are average values over three runs. Please refer to Appendix F.3, available in the online supplemental material for details of dataset and experimental settings.

QA can only handle QAP of size 8 due to the limitations in the D-Wave Chimera architecture. From Table 1 we know that QA is the fastest among all others. Table 2 shows the energy of the solution found by QA is the highest. This matches with the results of experiments in Section 3.3.

DA can directly handle up to QAP of size 90. \( T_{DA} \) 16 times faster than QBSolv on WH-90. In terms of the quality of the solutions, DA finds near-optimal solutions to WH-8. Although DA can handle a problem size of 90 directly
without decomposition heuristics, the $E_{DA}$ is distinctly higher than that of Tabu and SA. For problems larger than 90, we use decomposition heuristics to tackle the problem. $T_{decomp}$ in Table 1 is the time for DA solving all sub-problems. It scales linearly with the size of the problem. In terms of the quality of the solution, our decomposition heuristic can find a solution that is 14.3% lower than that of the random solutions on WH-180. However, as the number of partitions increases, the optimality against the random solution vanishes. For example, on WH-8100, there is no difference in energy between random solution and the decomposition heuristic.

Tabu is faster than all other methods on most datasets, except WH-8. In terms of the solution’s quality, both Tabu and SA can find better solutions than other methods. QBSolv did not find any feasible solution for WH-180.

### 5.2 Comparison on Warehouse Management

We also generate a set of warehouse problems to test the effectiveness of the decomposition heuristic in reducing warehouse travel distance. The input data distribution is perturbed to provide modified versions of datasets in Section 5.1. The name of the resulting problems has a tailing letter “b”. This is to match the findings in [28] that interaction between number of order variables in the QUBO form. QA performs poorly and SA can find better solutions than other methods. QBSolv did not find any feasible solution for WH-180.

### 6 DISCUSSION

#### 6.1 Quantum(-Inspired) Annealers: Pros, Cons and What’s Missing

**Speed.** The running time of D-Wave Quantum Annealer (QA) depends on the settings of its annealing process and is independent of the problem size. The time of Fujitsu Digital Annealer (DA) depends on the problem size. In our experiments, the problem-solving time of QA is generally fast on simple problem settings. DA has a good scalability in terms problem size.

To accelerate QA and DA from a problem perspective, we can reduce the size of the problem. For example, smart encoding technique [33] uses fewer qubits to represent larger problems for gate-based models. The equivalent research efforts in the domain of annealing-based computers is missing. On the other hand, problem simplification could also help. For example, data scaling [34] smoothens the energy landscape of a permutation optimisation problem and helps DA find promising results with fewer iterations. A more general simplification method is missing to facilitate solver for other combinatorial optimisation problems.

To accelerate QA and DA from a method perspective, we can optimise the annealing process for a quantum annealer [35] and a classical annealer [36]. This shortens the problem-solving time, as well as potentially improve of the optimality of the solutions. However, many works in this area are problem-specific and heavily parameterised. A general optimisation method is missing.

**Optimality.** Overall, both QA and DA have a decrease in the quality of solutions as the problem size increases, although the reason behind are totally different. QA suffers from more errors in quantum mechanics [37] and analog control [26] as the problem size increases. DA has difficulty in exploring the solution space as it expands quickly along with the increase of problem size.

QA outperforms DA in a simple problem setting, but loses its leading position when the problem setting gets complicated. Let $S_p$ denotes the solution space to the original problem and $S_o$ denotes the space expanded by the decision variables in the QUBO form. QA performs poorly

### TABLE 1

**QAP Time Comparison**

| Size      | 8      | 90     | 180    | 270    | 3600   | 8100   |
|-----------|--------|--------|--------|--------|--------|--------|
| $T_{Q}$   | 0.016  | -      | -      | -      | -      | -      |
| $T_{QBSolv}$ | 0.37   | 379.95 | 7454   | -      | -      | -      |
| $T_{DA}$  | 3.70   | 25     | -      | -      | -      | -      |
| $T_{decomp}$ | -     | 60     | 68     | 983    | 2080   | -      |
| $T_{Tabu}$| 0.02   | 0.08   | 0.15   | 0.19   | 23.5   | 64.3   |
| $T_{SA}$  | 683    | 556    | 716    | 594    | 3447   | 6884   |

### TABLE 2

**QAP Energy Comparison (Lower is Better)**

| Size      | 8      | 90     | 180    | 270    | 3600   | 8100   |
|-----------|--------|--------|--------|--------|--------|--------|
| $E_{Q}$   | 119    | -      | -      | -      | -      | -      |
| $E_{QBSolv}$ | 81     | 0.71m  | NA     | -      | -      | -      |
| $E_{DA}$  | 81     | 0.67m  | -      | -      | -      | -      |
| $E_{decomp}$ | -     | 2.21m  | 15.9m  | 12.96  | 115b   | -      |
| $E_{Tabu}$| 80     | 0.64m  | 1.95m  | 14.4m  | 10.96  | 103b   |
| $E_{SA}$  | 81     | 0.64m  | 1.96m  | 14.4m  | 10.66  | 91.7b  |
| $E_{Random}$ | -     | 0.75m  | 2.58m  | 17.5m  | 13.1b  | 115b   |

‘$m$’ stands for ‘$\times 10^6$’ while ‘$b$’ means ‘$\times 10^9$’.

### TABLE 3

**Warehous Travel Distance Comparison**

| Name       | ABC | COI | OOS | Random | decomp |
|------------|-----|-----|-----|--------|--------|
| WH-90b     | 733 | 876 | 773 | 741    | 765    |
| WH-180b    | 1418| 1434| 1425| 1418   | 1422   |
| WH-270b    | 3457| 4524| 3084| 3619   | 3098   |

**Index (COI) [30], class-based storage policy (ABC) [31], and representative assignment heuristics, i.e., Cube-per-order (COI) [30], class-based storage policy (ABC) [31], and Order-oriented-swapping (OOS) [32], as references. A brief review of these heuristics are available in Appendix F.1, which solves a block-structured QAP generated from WH-270b. The numbers in Table 3 are the travelling distance of a picker, given different warehouse assignment policies. We run the experiment five times and get the averages to indicate the performance of these policies.

Table 3 shows that OOS is the best among the first three heuristics. The performance of our heuristics is very close to that of OOS. The objective function of QAP and the cost of warehouse assignment are not identical. Our decomposition heuristic does not perform as well as direct methods without decomposition from the point of view of QAP, but it performs well on warehouse assignment problems.
when $S_p$ is smaller than $S_\alpha$. In comparison, DA is less sensitive to this factor. For example, QA outperforms DA on max-cut in Section 3.1.2, but underperforms DA on MVC in Section 3.2.1, despite the two experiments share the same dataset. We reach this conclusion by comparing QA and DA on different problems settings, which is missing in existing benchmark works.

There has been quite a lot of research in improving annealing-based computers. Techniques for improving speed from a problem perspective could also improve the optimality. Error Correction [37], [38] is a popular research direction for QA, but not DA. Furthermore, finetuning hyper-parameters such as the annealing process [35], or penalty weight [39] can also potentially improve its optimality.

### 6.2 Decomposition on QAP

Decomposition improves the scalability of annealing-based computers. Experiments in Section 5 suggest our heuristic solves block-structural QAP with improved quality when the size is limited below 270. However, for larger problem sizes, our heuristic gradually loses its utility.

*Cases Where Decomposition Fail.* There could be a few reasons. First, solutions to sub-QAPs are not optimal. Second, the heuristic may be inaccurate as size increases. It is possible that when too many columns of 90 are stacked together, the effect of assigning item pairs in one column is superseded by the vast shape of the warehouse, such that even items from different columns, which have relatively low interaction frequency, contribute significantly to the overall traffic because the distance becomes large. Therefore, it is not advisable to split items and locations into too many subsets.

*Matching the Subsets.* In Section 4.2, the heuristic partitions the items and locations according to an aggregate interaction-frequency measure and distance measure. This is a special case of the more generalised randomised decomposition technique in [21], in which the items and locations are randomly taken to form sub-QAPs and lead to promising results. Therefore, one of the possibilities is to add a random layer on top of the current decomposition layer. In a similar spirit, the subsets are matched randomly in this project. Instead, there could be other organised ways as mentioned in Section 4.2. When the number of bijections is small, the matching can be considered exhaustive to determine which one is the best. For example, for WH-180, there are only 2 item and location subsets. There are only 2 ways to match the subsets. A more interesting but much more difficult question is how a matching would be a good seed.

The overall decomposition heuristic is only proven to sustain optimality for a very restricted simplification. It would be encouraging if further advancements can be made in this direction, expanding the solution quality guarantee to a larger class of block-structural QAPs. However, it is so far unclear how that could be done because that would involve analysing the distance matrix case-by-case.

### 7 Conclusion

Recent quantum-(inspired) annealers show promise in solving combinatorial optimisation problems. In this paper, we compared a true quantum annealer with a CMOS digital annealer on three problems ranging from simple to complex ones. We also compare them in the context of decomposition. Experiments suggest that the performance of quantum-(inspired) annealers is closely related to the problem settings. Decomposition techniques extend the scalability of quantum-(inspired) annealers. However, getting promising solutions is still challenging for computing devices with limited capability. Through experiments and analysis, we highlighted the research directions that can improve the utility and scalability of quantum-(inspired) annealers.

### References

[1] F. Glover, G. Kochenberger, and Y. Du, “A tutorial on formulating and using qubo models,” 2018, arXiv:1811.11538.

[2] T. Kadowaki and H. Nishimori, “Quantum annealing in the transverse ising model,” Phys. Rev. E, vol. 58, no. 5, 1998, Art. no. 5355.

[3] S. Poljak and Z. Tuza, “Maximum cuts and large bipartite subgraphs,” DIMACS Ser., vol. 20, pp. 181–244, 1995.

[4] Z. Bian, F. Chudak, W. Macready, A. Roy, R. Sebastiani, and S. Vicari, “Solving the quadratic assignment problem with a quantum annealer: Foundations and a preliminary report,” in Proc. Int. Symp. Front. Combining Syst., 2017, pp. 153–171.

[5] E. Grant, T. S. Humble, and B. Stump, “Benchmarking quantum annealing controls with portfolio optimization,” Phys. Rev. Appl., vol. 15, no. 1, pp. 014012, 2021.

[6] D. Inoue, A. Okada, T. Matsumori, K. Aihara, and H. Yoshida, “Traffic signal optimization on a square lattice using the d-wave quantum annealer,” 2020, arXiv:2003.07527.

[7] M. S. Sao, H. Watanabe, Y. Musha, and A. Usunomiya, “Application of digital annealer for faster combinatorial optimization,” Fujitsu Sci. Tech. J., vol. 55, no. 2, pp. 45–51, 2019.

[8] D. Vert, R. Sirdey, and S. Louise, “On the limitations of the chimera graph topology in using analog quantum computers,” in Proc. Int. Symp. Front. Combining Syst., 2019, pp. 226–229.

[9] D. W. S. Inc. “Quantum annealing Topologies,” 2022. Accessed: Sep. 01, 2021, [Online]. Available: https://docs.dwavesys.com/docs/latest/c_gs.4.html

[10] M. Aramon, G. Rosenberg, E. Vialante, T. Miyazawa, H. Tamura, and H. G. Katzgraber, “Physics-inspired optimization for quantum unconstrained problems using a digital annealer,” Front. Phys., vol. 7, 2019, Art. no. 48.

[11] Z. Baghsh, M. J. Mahabadi, S. Valae, “Digitally annealed solution for the maximum clique problem with critical application in cellular v2x,” in Proc. IEEE Int. Conf. Commun., 2019, pp. 1–7.

[12] P. Miasnikof, S. Hong, and Y. Lawryshyn, “Graph clustering via QUBO and digital annealing,” 2020, arXiv:2003.03872.

[13] A. Maruo, H. Igarashi, H. Oshima, and S. Shimokawa, “Optimization of planar magnet array using digital annealer,” IEEE Trans. Magn., vol. 56, no. 3, pp. 1–7, Mar. 2020.

[14] G. Bass, M. Henderson, J. Heath, and J. Dunly, “Optimizing the optimizer: Decomposition techniques for quantum annealing,” Quantum Mach. Intell., vol. 3, no. 1, pp. 1–14, 2021.

[15] M. Ohzeki, A. Miki, M. J. Miyama, and M. Terabe, “Control of automated guided vehicles without collision by quantum annealer and digital devices,” Front. Comput. Sci., vol. 1, 2019, Art. no. 9.

[16] 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, 2018, Art. no. 031016.

[17] O. Seker, N. Tanoumand, and M. Bodur, “Digital annealer for quadratic unconstrained binary optimization: A comparative performance analysis,” 2020, arXiv:2012.12264.

[18] M. Kowalski, T. Albash, J. Hen, and D. A. Lidar, “3-regular 3-XORSAT planted solutions benchmark of classical and quantum heuristics,” Quantum Sci. Technol., vol. 7, 2020, Art. no. 025008.

[19] M. Booth, S. P. Reinhardt, and A. Roy, “Partitioning optimization problems for hybrid classical,” Quantum Computing Company, Tech. Rep. 14-1006A-A, 2017.

[20] G. Bass, M. P. Henderson, J. Heath, and J. Dunly, “Optimizing the optimizer: Decomposition techniques for quantum annealing,” Quantum Mach. Intell., vol. 3, pp. 1–14, 2020.

[21] K. Mihić, K. Ryan, and A. Wood, “Randomized decomposition solver with the quadratic assignment problem as a case study,” INFORMS J. Comput., vol. 30, no. 2, pp. 295–308, 2018.
[22] T. Stollenwerk, E. Lobe, and M. Jung, “Flight gate assignment with a quantum annealer,” in International Workshop Quantum Technol. Optim. Problems, vol. 2018, pp. 99–110, 2019.

[23] Z. Yang and M. J. Dinneen, Graph minor embeddings for D-wave computer architecture, Centre for Discrete Mathematics and Theoretical Computer Science, Tech. Rep. CDMTCS-503, 2016.

[24] D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner, Graph Partitioning and Graph Clustering, Providence, RI, USA: American Mathematical Society, 2013, vol. 588.

[25] R. E. Burkard, S. E. Karisch, and F. Rendl, “QAPLIB—a quadratic assignment problem library,” J. Glob. Optim., vol. 10, no. 4, pp. 391–403, 1997.

[26] A. Pearson, A. Mishra, I. Hen, and D. A. Lidar, “Analog errors in quantum annealing: Doom and hope,” NPJ Quantum Inf., vol. 5, pp. 1–9, 2019.

[27] A. Misevicius, “A tabu search algorithm for the quadratic assignment problem,” Comput. Optim. Appl., vol. 30, no. 1, pp. 95–111, 2005.

[28] M. T. Tsige, “Improving order-picking efficiency via storage assignments strategies,” Master’s thesis, Industrial Engineering & Management, Univ. Twente, The Netherlands, Feb. 2013.

[29] H. Ruijter, “Improved storage in a book warehouse: Design of an efficient tool for slotting the manual picking area at wolters-noordhoff,” Master’s thesis, Fac. Behav., Manage. Social Sci., Univ. Twente, 2007.

[30] C. J. Malmborg and K. Bhaskaran, “A revised proof of optimality for the cube-per-order index rule for stored item location,” Int. J. Phys. Distrib. Logistics Manag., vol. 34, no. 7, pp. 534–544, 2004.

[31] R. J. Mantel, P. C. Schuur, and S. S. Heragu, “Order oriented slotting: A new assignment strategy for warehouses,” Eur. J. Ind. Eng., vol. 1, no. 3, pp. 301–316, 2007.

[32] B. Tan, M.-A. Lemonde, S. Thanasilp, J. Tangpanitanon, and D. G. R. J. Mantel, P. C. Schuur, and S. S. Heragu, “Order oriented slotting: A new assignment strategy for warehouses,” Eur. J. Ind. Eng., vol. 1, no. 3, pp. 301–316, 2007.

[33] B. Tan, M.-A. Lemonde, S. Thanasilp, J. Tangpanitanon, and D. G. Angelakis, “Qubit-efficient encoding schemes for binary optimisation problems,” Quantum, vol. 5, 2021, Art. no. 454.

[34] S. T. Goh, S. Gopalakrishnan, J. Bo, and H. C. Lau, “A hybrid framework using a qubo solver for permutation-based combinatorial optimization,” 2020, arXiv:2005.12767.

[35] D. Venturelli and A. Kondratyev, “Reverse quantum annealing approach to portfolio optimization problems,” Quantum Mach. Intell., vol. 1, no. 1, pp. 17–30, 2019.

[36] S. V. Isakov, I. N. Zhintchenko, T. F. Rønnow, and M. Troyer, “Optimised simulated annealing for ising spin glasses,” Comput. Phys. Commun., vol. 192, pp. 265–271, 2015.

[37] K. L. Rudzien, T. Albash, and D. A. Lidar, “Error-corrected quantum annealing with hundreds of qubits,” Nature Commun., vol. 5, no. 1, pp. 1–10, 2014.

[38] W. Vinci, T. Albash, and D. A. Lidar, “Nested quantum annealing correction,” NPJ Quantum Inf., vol. 2, no. 1, pp. 1–6, 2016.

[39] T. Huang, S. T. Goh, S. Gopalakrishnan, T. Luo, Q. Li, and H. C. Lau, “QROSS: QUBO relaxation parameter optimisation via learning solver surrogates,” in Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. Workshops, 2021, pp. 35–40.

Tian Huang received the BSc and PhD degrees from the Shanghai Jiao Tong University, Shanghai, China, in 2008 and 2016, respectively. He is currently a research scientist with the Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore. His research interests lie in high-performance computing and machine learning algorithms.

Jun Xu received the bachelor’s degree with Honours (Highest Distinction) in computer engineering from National University of Singapore (NUS), in 2020. He is currently working toward the master’s degree in mathematics with National University of Singapore. From 2019 to 2020, he was the main contributor to the quantum annealing project at IHPC, A*STAR, Singapore. In the project, he explored the use cases of both quantum and digital annealers on the solution of quadratic assignment problem.

Tao Luo received the bachelor’s degree from the Harbin Institute of Technology, Harbin, China, in 2010, the master’s degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2013, and the PhD degree from the School of Computer Science and Engineering, Nanyang Technological University, Singapore, in 2018. He is currently a research scientist with the Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore. His current research interests include high-performance computing, quantum computing, reconfigurable computing system, hardware–software co-exploration, and efficient artificial intelligence and its application.

Xiaoze Gu received the PhD degree in computer science and engineering from Nanyang Technological University, Singapore, in 2018. He is currently a postdoctoral researcher with the Future Network of Intelligence Institute, the Chinese University of Hong Kong, Shenzhen. His research interests include real-time system, scheduling algorithm and machine learning.

Rick Goh received the PhD degree in electrical and computer engineering from the National University of Singapore, Singapore, in 2006. He is the director of the Computing Science Department, A*STAR Institute of High Performance Computing (IHPC), Singapore. The research focus areas include artificial intelligence (AI), high-performance computing (HPC), distributed computing, machine and deep learning, complex systems, human-machine interaction, and modeling and simulation.

Weng-Fai Wong (Senior Member, IEEE) received the BSc degree from the National University of Singapore, in 1988, and the DrEngSc degree from University of Tsukuba, Japan, in 1993. He is currently an associate professor with the Department of Computer Science, National University of Singapore. His research interests include computer architecture, compilers, and high-performance computing.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.