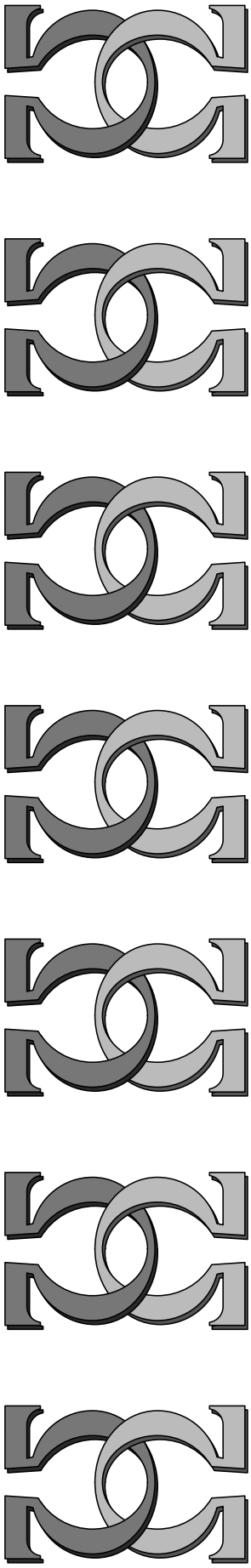


CDMTCS Research Report Series

A Hybrid Quantum-Classical Paradigm to Mitigate Embedding Costs in Quantum Annealing

A. A. Abbott[1] C. S. Calude[2], M. J. Dinneen [2], R. Hua[2]

[1]University Grenoble Alpes, CNRS, France
[2]University of Auckland, New Zealand

CDMTCS-520
February 2018

Centre for Discrete Mathematics and
Theoretical Computer Science

A Hybrid Quantum-Classical Paradigm to Mitigate Embedding Costs in Quantum Annealing

Alastair A. Abbott[*], Cristian S. Calude[†], Michael J. Dinneen[†] and Richard Hua[†]

March 6, 2018

Abstract

Despite rapid recent progress towards the development of quantum computers capable of providing computational advantages over classical computers, it seems likely that such computers will, initially at least, be required to run in a hybrid quantum-classical regime. This realisation has led to interest in hybrid quantum-classical algorithms allowing, for example, quantum computers to solve large problems despite having very limited numbers of qubits. Here we propose a hybrid paradigm for quantum annealers with the goal of mitigating a different limitation of such devices: the need to embed problem instances within the (often highly restricted) connectivity graph of the annealer. This embedding process can be costly to perform and may destroy any computational speedup. We will show how a raw speedup that is negated by the embedding time can nonetheless be exploited to give a practical speedup to certain computational problems. Our approach is applicable to problems in which embeddings can be used multiple times to solve related problems, and may allow quantum speedups to be more readily exploited. As a proof-of-concept we present an in-depth case study of a problem based on the maximum weight independent set problem. Although we do not observe a quantum speedup experimentally, the advantage of the hybrid approach is robustly verified, showing how a potential quantum speedup may be exploited.

1 Introduction

Quantum computation has the potential to revolutionise computer science, and as a consequence has, since its inception, received a great deal of attention from theorists and experimentalists alike. Although much progress has been made through the concerted efforts of the community, we are still some distance from being able to build sufficiently large-scale universal quantum computers to realise this potential [1, 2].

More recently, however, significant progress has been made in the development of special-purpose quantum computers. This has been driven by the realisation that, by dropping the requirement of being able to efficiently simulate arbitrary computations and relaxing some of the constraints that make large-scale universal quantum computing difficult (e.g., the ability to apply gates to arbitrary pairs of, possibly non-adjacent, qubits), such devices can be more easily engineered and scaled. The expectation is that with this approach one may be able to exploit some of the capabilities of quantum computation—even if its full abilities are for now beyond our reach—to obtain lesser, but nevertheless practical, advantages in real-world applications. Quantum annealers, which solve particular optimisation problems, exemplify this approach, and significant progress has been made in recent years towards engineering moderately large-scale such devices [3, 4]. This approach has been pursued particularly zealously by D-Wave, who have developed

[*]University Grenoble Alpes, CNRS, Grenoble INP, Institut Néel, 38000 Grenoble, France
[†]Department of Computer Science, University of Auckland, Private Bag 92019, Auckland, New Zealand

quantum annealers with upwards of 2000 qubits (e.g., the D-Wave 2000Q™ machine [5]), and and are thus of sufficient size to tackle problems for which their performance can meaningfully be compared to classical computational approaches.

In this paradigm, however, it is non-trivial to compare the performance of quantum solutions to classical ones, since the focus is on obtaining real-world gains in domains where heuristics tend to be at the core of the best classical approaches. Indeed, this issue is at the heart of recent debate surrounding the performance of D-Wave machines [6, 7]. In particular, instead of focusing on asymptotic analyses, one must compare the performance of classical and quantum devices empirically. But performing benchmarks fairly is difficult, especially when there is often debate as to which classical algorithm should be taken for comparison [8–11]. This is further complicated by the crucial realisation that such special-purpose quantum devices are operated in a fundamentally different way to the classical ones which they must be compared with: typically, they operate in conjunction with a non-trivial pipeline of classical pre- and post-processing whose contribution is far from negligible on the performance of the device, and may even be the difference between obtaining a quantum speed-up or not. Note that such pre- and post-processing costs may arise when generic classical solvers (e.g. Integer Programming or SAT solvers) are used for optimisation problems, and although such solvers may not be the fastest classical algorithms they are nonetheless of much practical interest and, when used, this processing pipeline should similarly be taken into account.

In this paper, motivated by the need to take into account the cost of classical processing in benchmarking quantum annealers, we propose a hybrid quantum-classical approach for developing algorithms that mitigates the cost of this processing. In particular, focusing on D-Wave's quantum annealers, where this processing involves a costly classical "embedding" stage, which maps an arbitrary problem instance into one compatible with D-Wave's limited connectivity constraints, we formulate a generic hybrid approach that mitigates this cost allowing any advantage present to be accessed more directly [12]. The embedding problem is time-consuming (more precisely, NP-hard), and experimental studies indicate that it's quality can have strong effects on performance [13, 14]. A similar type of approach has previously been suggested as a theoretical means to exploiting Grover's algorithm [15], and differs from recent hybrid approaches for quantum annealing [16–19] and computing [20, 21] that instead aim to provide quantum advantages in situations where far fewer qubits are available than would be needed to execute a complete quantum algorithm for the problem in question.

Having outlined this generic framework for hybrid computing with D-Wave, we then present a hybrid algorithm that is based around a D-Wave solution to the maximum-weight independent set (MWIS) problem. Although the problem this algorithm solves, called the dynamically weighted MWIS problem, perhaps has limited independent interest, it serves as a strong proof-of-concept for more complex algorithms, and we reinforce this by implementing it experimentally on a D-Wave 2X machine [22]. The results of the experiment show a large improvement of the hybrid algorithm over a standard quantum annealing approach, in which the embedding process is naively repeated many times. We further compare the hybrid algorithm to a standard classical algorithm. Although we do not observe an overall speedup using the hybrid algorithm, the scaling behaviour of this approach compares favourably to that of the classical algorithm, leaving open the possibility of future speedups for this problem.

2 D-Wave's quantum annealing framework

2.1 Quantum annealing

Quantum annealing is a finite temperature implementation of adiabatic quantum computing [23], in which the optimisation problem to be solved is encoded into a Hamiltonian H_p (the quantum operator corresponding to the system's energy) such that the ground state(s) of H_p correspond(s) precisely to the solution(s) to the problem (of which there may be several). The computer is initially prepared in the ground state of a Hamiltonian H_i, which is then slowly evolved into the target Hamiltonian H_p. This computation can be described by the time-dependent Hamiltonian $H(t) = A(t)H_i + B(t)H_p$ for $0 \leq t \leq T$, where $A(0) = B(T) = 1$ and $A(T) = B(0) = 0$. T is called the *annealing time* and the functions A and B determine the *annealing schedule*, which for a D-Wave machine are such that the evolution is close to a linear transition from H_i to H_p [3, 24].

If the computation is performed sufficiently slowly, the Adiabatic Theorem guarantees that the system will remain in a ground state of H_p throughout the computation and the final state will thus correspond to an optimal solution to the problem at hand [23]. In the ideal adiabatic limit, the time required for such a computation scales as the inverse-square of the minimum spectral-gap[1] (i.e., the minimum difference between the ground and first excited states of $H(t)$). However, in the finite temperature regime of quantum annealing, a trade-off must be found between evolving the system sufficiently slowly and avoiding the perturbing affect of the environment. As a consequence, the final state is only a correct solution with a certain probability, and the (hence probabilistic) computation must be repeated many times to obtain the desired solution (or a sufficiently close approximation thereof) [3, 26].

2.2 Quadratic unconstrained Boolean optimisation

Although the adiabatic computational model is universal [27], the recent success of quantum annealing has come about by focusing on implementing specific types of Hamiltonians that are simpler to engineer and control, despite the fact they might not be capable of efficiently simulating arbitrary quantum circuits. In particular, D-Wave's devices can be modelled by a two-dimensional Ising spin glass Hamiltonian, and it is thus capable of solving the *Ising spin minimisation problem*, a well-known NP-hard optimisation problem [3, 28]. This problem is equivalent, via a simple mapping of spin values (± 1) to bits (0 or 1), to the *Quadratic Unconstrained Boolean Optimisation (QUBO) problem* [29]. In this paper we will use this formulation, as it will allow us to represent in detail a little more compactly the algorithms.

The QUBO problem is the task of finding the input $\mathbf{x}^*$ that minimises a quadratic objective function of the form $f(\mathbf{x}) = \mathbf{x}^T Q \mathbf{x}$, where $\mathbf{x} = (x_1, \ldots, x_n)$ is a vector of n binary variables and Q is an upper-triangular $n \times n$ matrix of real numbers:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \mathbf{x}^T Q \mathbf{x} = \arg\min_{\mathbf{x}} \sum_{i \leq j} x_i Q_{(i,j)} x_j, \text{ where } x_i \in \{0, 1\}. \tag{1}$$

Note that arbitrary quadratic objective functions g can be converted to this form. Since $x_i^2 = x_i$ for $x_i = 0$ or 1, linear terms of g can be encoded as the diagonal entries of a Q for f. Furthermore, any constant terms in g can be ignored since they do not affect the objective minimisation with respect to $\mathbf{x}$.

In the quantum annealing model of the QUBO problem, each x_i corresponds to a qubit while Q defines the problem Hamiltonian H_p. Specifically, the diagonal terms $Q_{(i,i)}$ represent the local

[1] Determining the minimum spectral gap, and thus the time required for computation, is unfortunately itself a computationally difficult problem [25].

field applied to each qubit, whereas the non-negative off-diagonal terms $Q_{(i,j)}$, $i < j$, correspond to a coupling between qubits x_i and x_j. For a given QUBO problem Q, these couplings may be conveniently represented as a graph $G_L = (V_L, E_L)$ representing the interaction between qubits, where $V_L = \{1, \ldots, n\}$ is the set of qubits and $E_L = \{\{i, j\} \mid Q_{(i,j)} \neq 0,\ i < j\}$ are the edges representing the couplings between qubits. For reasons that will soon be apparent, we will refer to such a graph for a given QUBO problem as the *logical graph*, and the set of qubits the QUBO problem is represented over the *logical qubits*.

2.3 Hardware constraints and embeddings

The comparative ease in engineering devices which naturally solve the QUBO problem has been crucial for the recent experimental success of quantum annealing. This restriction to solving natively specific NP-hard problems is nonetheless not the only compromise associated with the engineering constraints of such devices, even if it is the most immediately obvious. More subtly, it remains exceedingly difficult to control interactions between qubits that are not physically near to one another, and as a result it is not possible to implement directly any instance of the QUBO problem: this would require directly coupling arbitrary pairs of qubits, which is currently infeasible. Instead, the couplings possible on a quantum annealer are specified by a graph $G_P = (V_P, E_P)$, where V_P is the set of qubits on the device, and an edge $\{i, j\} \in E_P$ signifies that qubits i and j can be physically coupled. The graph G_P is called the *physical graph*, and the qubits V_P are the *physical qubits* [29, 30].

The physical graphs implemented on D-Wave's devices are *Chimera graphs* χ_k, which are $k \times k$ grids of $K_{4,4}$ graphs, with connections between adjacent 'blocks' as shown in Figure 1.[2] Specifically, each qubit is coupled with 4 other qubits in the same $K_{4,4}$ block and 2 qubits in adjacent blocks (except for qubits in blocks on the edge of the grid, which are coupled to a single other block). A more formal definition of the Chimera graph structure can be found in [32], but will not be necessary for our purposes.

The Chimera graph is, crucially, relatively sparse and near-to-planar, with qubits separated by paths of length no longer than $2k$. Although the specific choice of hardware graph is an engineering decision and may conceivably be changed in future devices, any alternative physical graph is likely to have similar properties since the tradeoff between connectivity and practicability is a core feature (and intrinsic limitation) of the current approach to quantum annealing [30, 33]. It is therefore essential to take into account these limitations of the hardware graph in any approach to solving problems with quantum annealers.

Since the logical graph G_L for a QUBO problem instance Q will not, in general, be a subgraph of the physical graph $G_P = \chi_k$, the problem instance on G_L must be mapped to an equivalent one on G_P. This process involves two steps: first, G_L must be *embedded* in G_P, and secondly the weights of the QUBO problem (i.e., the non-zero entries in Q) must be adjusted so that valid solutions on G_P are mapped to valid solutions on G_L.

The embedding stage amounts to finding a *minor embedding* of $G_L = (V_L, E_L)$ into $G_P = (V_P, E_P)$ [29, 34], i.e., an embedding function $f : V_L \to 2^{V_P}$ such that

1. the sets of vertices $\{f(v) \mid v \in V_L\}$ are disjoint;
2. for all $v \in V_L$, there is a subset of edges $E' \subseteq E_P$ such that $G' = (f(v), E')$ is connected;

[2] It is possible to define a more general family of $n \times m \times L$ Chimera graphs that are $n \times m$ grids of $K_{L,L}$ graphs, as in [31]. However, all devices to date have been square grids of $K_{4,4}$ graphs and, so that, in order to talk more precisely about scaling behaviour, we adopt the convention of fixing $L = 4$ and $n = m$ [25]. This is further justified by noting that increasing L involves increasing the *density* of qubit couplings, which is technically much more difficult than increasing the grid size.

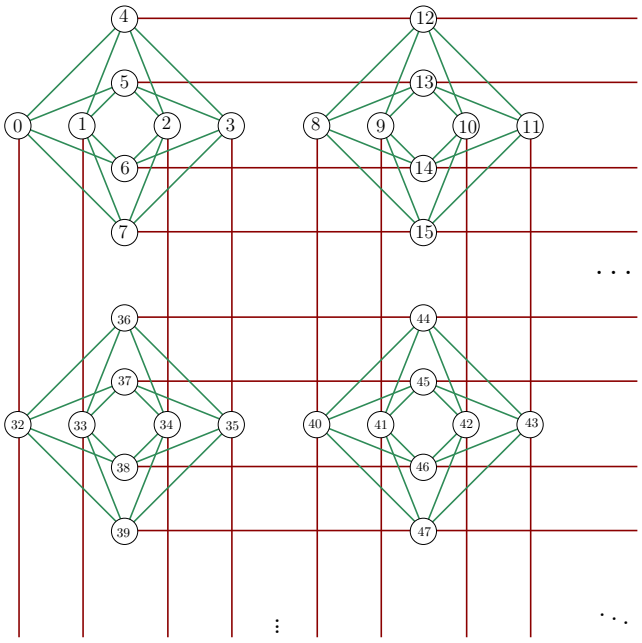


Figure 1: A portion of a Chimera graph, showing four $K_{4,4}$ blocks. In general, the graph χ_k consists of a $k \times k$ grid of such blocks, with connections between adjacent blocks as shown.

3. if $\{u, v\} \in E_L$, then there exist $u', v' \in V_P$ such that $u' \in f(u)$, $v' \in f(v)$ and $\{u', v'\}$ is an edge in E_P.

Typically, this involves mapping each logical qubit to "chains" or "blocks" of physical qubits. In general, a QUBO instance using n logical qubits will require up to $O(n^2)$ physical qubits since the smallest Chimera graph in which the complete graph K_{4k} can be embedded in is χ_k, requiring $4k(k+1)$ physical qubits [25, 33]. The embedding thus already entails, in general, a quadratic increase in problem size which needs to be taken into account when benchmarking quantum annealers.

The problem of finding a minor embedding is itself computationally difficult [29]. Of course, if one has sufficiently many physical qubits to embed K_n then any n-qubit logical graph can trivially be embedded into the physical graph. However, this trivial embedding is generally rather wasteful in that it uses many more qubits that are actually needed, and qubits are precious resources as the practical limits of quantum annealing are being pushed to try and find real-world speedups. Perhaps more importantly, the difficulty of a problem for quantum annealers increases with the number of physical qubits, so even when such a naive embedding exists there may be a significant advantage in looking for smaller embeddings (the feasibility of a problem may even depend on it). The embedding process may thus, in light of its computational difficulty, contribute significantly to the time required to solve a problem in practice. Currently, the standard approach to finding such an embedding is to use heuristic algorithms (see, e.g., [35]).

The second stage, which ensures that the validity of solutions is preserved, involves deciding on how to share the weights associated with each logical qubit v between the physical qubits $f(v)$ it it is mapped to. Since the weights must all fall within a finite range[3] and there is a limited analogue precision with which the weights can be set, this process can effectively amplify the relative effects of such errors and thus decrease the probability of finding the correct solution [25, 29, 36]. This stage thus further exemplifies the need to avoid unnecessarily large embeddings, but does not have the same intrinsic computational cost as the embedding process proper.

[3]Physically, the quantum annealer requires that the QUBO weights satisfy $|Q_{(i,j)}| \leq 1$ for all i and j. An arbitrary problem specified by Q must thus be scaled to satisfy this constraint.

2.4 Benchmarking quantum annealers

Although from a theoretical perspective it is expected that general purpose quantum computers will provide a computational advantage over classical algorithms, there has been much debate over whether or not quantum annealing provides any such speedup in practice [8, 12, 30]. While there are many reasons for this debate, much of it has stemmed from disagreement over what exactly constitutes a quantum "speedup" and, indeed, how to determine if there is one [8]. In this paper we will focus primarily on the run-time performance in investigating whether a quantum speedup is present, rather than the (empirically estimated) scaling performance of quantum algorithms.

One of the key points complicating this issue is the fact that, even in the standard circuit model of quantum computation, it is not generally believed that an exponential speedup is possible for NP-hard problems such as the QUBO problem [37]. Leading quantum algorithms instead typically provide a quadratic or low-order polynomial speedup [38]. In practice, heuristic algorithms are generally used to solve such optimisation problems and the probabilistic nature of quantum annealing means that it is also best viewed in this light [8, 25]. This means that, rather than theoretical algorithmic analysis, empirical measures are essential in benchmarking quantum annealing against classical approaches.

2.4.1 Measuring the processing time

Good benchmarking will, first of all, need to make use of fair and comprehensive metrics to determine the running time of both classical and quantum algorithms for a problem. In particular these need to properly take into account not only the "wall-time" of different stages of the quantum algorithm, but also its probabilistic nature. To understand how this can be done, we first need to outline the different stages of the quantum annealing process [25].

1. *Programming:* The problem instance is loaded onto the annealing chip (QPU), which takes time t_{prog}.
2. *Annealing:* The quantum annealing process is performed and then the physical qubits are measured to obtain a solution; this takes time[4] t_a.
3. *Repetition:* Step 2 is performed k times to obtain k potential solutions.

The *quantum processing time* (QPT) is thus

$$t_{\text{proc}} = t_{\text{prog}} + k\, t_a.$$

For any given run of a quantum annealer, there is a non-zero probability of obtaining a correct solution to the problem at hand, which depends on both the annealing time t_a and the number of repetitions k. Moreover, for any specific problem instance, the optimal values of these parameters are not known *a priori*, so the performance of a quantum annealing algorithm will be determined by the optimal values of these parameters for the hardest problems of a given size n [8]. On current devices, however, the minimal annealing time of 20μs has repeatedly been found to be longer than the optimal time [8, 25, 39, 40].

With these considerations on hand, a relatively fair and robust way to measure the quantum processing time is the "time to solution" (TTS) metric [8, 41], which is based on the expected number of repetitions needed to obtain a valid solution with probability p (one typically takes

[4]Note that this is sometimes referred to as the "wall clock time" in the literature. For simplicity, we choose to englobe all times associated with an annealing cycle (e.g. readout and inter-sample thermalisation times) along with the annealing time *per se* into t_a.

$p = 0.99$).[5] If the probability per annealing sample of obtaining a solution is s (which can be estimated empirically), then this is calculated as

$$k_{99} = \frac{\log(1-p)}{\log(1-s)}, \tag{2}$$

and the quantum processing time is thus calculated with this k as $t_{\text{proc}} = t_{\text{prog}} + k_{99}\, t_a$.

In practice, unfortunately, even for moderate problem sizes, quantum annealing (and, indeed, classical annealing) simply does not find a correct solution to many problem instances [8, 41, 42]. Thus, although no worst case running time for such problems can be calculated, it is often instructive to look at the QPT for restricted classes of problems of particular interest or of limited difficulty. In particular, several authors have applied this to difficulty "quantiles", calculating the QPT for, e.g., the 75% of problems that can be solved the quickest. Investigating how the QPT scales with problem difficulty in this way permits some comparison with classical algorithms where it would otherwise be difficult or even impossible [8, 41].

Existing investigations have primarily focused on comparing directly the QPT with the processing time of a classical algorithm in order to look for what we call a "raw quantum speedup". However, it is essential to realise that the time used by the QPU and measured by the QPT refers only to a subset of the processing required to solve a given problem instance using a quantum annealer. Specifically, a complete quantum algorithm for a problem instance P involves, as a minimum requirement, the following steps:

1. *Conversion:* The problem instance P must be converted into a QUBO instance $Q(P)$, typically via a polynomial-time reduction taking time t_{conv}.
2. *Embedding:* The QUBO problem $Q(P)$ must be embedded into the Chimera hardware graph taking time t_{embed}.
3. *Pre-processing:* The embedded problem is pre-processed, which involves calculating (appropriately scaled) weights for the embedded QUBO problem, taking time t_{pre}.
4. *Quantum processing:* The annealing process is performed on the QPU, taking time t_{proc}.
5. *Post-processing:* The samples are post-processed to choose the best candidate solution, check its validity, and perform any other post-processing methods to improve the solution quality[6] [25, 36] taking time t_{post}. The QUBO solution must finally be converted back to a solution for the original problem P.

The total processing time is thus[7]

$$T_Q = t_{\text{conv}} + t_{\text{embed}} + t_{\text{pre}} + t_{\text{proc}} + t_{\text{post}}. \tag{3}$$

The realisation that these other steps must be included in the analysis is emphasised by the fact that in practical problems the embedding time often dominates the time used by the annealer itself. Previous investigations have largely avoided this by focusing on artificial problems

[5] It is possible to generalise the TTS method to a time-to-target (TTT) method [9], where one is interested in the expected time to obtain a solution that is sufficiently good with respect to some (perhaps problem-dependent) measure. Although this approach is likely to be very useful in benchmarking larger real-world problems, we focus on the TTS approach here (which can be seen as a specific case of TTT).

[6] On D-Wave's annealer, for example, a local search may optimally be performed to improve the solution quality. The k repetitions that are performed in the quantum processing step are broken into fixed "batches" of k/b samples (where b depends on the problem but not on k) and batches are post-processed in parallel with the annealing of the following one; this justifies the consideration of this post-processing as contributing towards the constant overhead t_{post}, as only the post-processing of the final batch contributes to T_Q. Note that such post-processing already constitutes a form of hybrid quantum-classical approach.

[7] As a convention, we will use lower case letters t for the timings of subtasks, and upper case T's to denote overall times of computation.

"planted" in the Chimera graph so that no embedding is necessary [6, 8, 39, 41, 42]. Although finding a raw speedup in such situations is clearly a necessary condition for a quantum speedup, it does not guarantee that any corresponding speedup will carry over into practical problems.

It is therefore the time T_Q which should be used in a fair comparison with classical algorithms. Note that this still makes use of the TTS approach discussed above, except one must now take into account the tradeoff between the quality of an embedding and the time spent finding it in order to determine the optimal annealing parameters.

2.4.2 Comparing classical and quantum algorithms

To properly benchmark quantum annealing against classical algorithms it is necessary not only to have fair measures of the cost of obtaining a solution, but one must also compare fairly the quantum annealer to a suitable classical algorithm.

Much of the controversy regarding potential speedups with quantum annealing has been due to the fact that the performance of quantum annealing has been compared to that of simulated annealing or simulated quantum annealing. Although such studies certainly have merit, since they allow one to investigate whether quantum annealing is indeed behaving as expected [6, 41, 42] and such a speedup is at least a necessary condition for a real quantum speedup, it has repeatedly been pointed out that, in such cases, classical annealing techniques are generally far from optimal and any observed speedups have disappeared when better classical algorithms were used [42]. In [8], this type of quantum speedup has been termed a "limited speedup".

Ideally, comparison should be made against the best classical algorithm for a problem. In practice, such an algorithm is rarely, if ever, known, especially for problems where heuristics dominate, and certain algorithms may perform better on certain subsets of problems. The best one can do in practice, then, is to look for a "potential quantum speedup" [8] by comparing against the best available classical algorithm for the problem at hand.

Finally, it is important to make sure the performance measures for both quantum and classical algorithms are compatible. That is, the classical processing time T_C should be calculated using a TTS metric as for T_Q (if the classical algorithm is deterministic, this simply reduces to the computation time), and should include all aspects of the classical computation, including pre- and post-processing and reading input. Note that by including the cost of embedding in the quantum and classical processing times, we make sure that what we calculate is a function of the problem size n and not the number of physical qubits.

3 Hybrid quantum-classical computing

As we discussed in the previous section, most of the effort in determining whether or not quantum annealing can, in practice, provide a computational speedup has focused purely on determining the existence of a *raw quantum speedup*, which does not take into account the associated classical processing that is inseparable from a quantum annealer. Such a raw speedup is certainly a necessary condition for practical quantum computational gains, and its study is therefore well justified. However, even if there is a raw speedup there are many reasons why this might not translate into a *practical* quantum speedup.

A practical speedup is possible for a problem if we are able to give a quantum algorithm such that $T_Q < T_C$, where (we recall) T_C is the classical processing time for the best available classical algorithm for the problem. From the definition of T_Q in (3), it is clear than, even if $t_{\text{proc}} < T_C$, the conversion, embedding and pre/post-processing may provide obstructions to obtaining a

practical speedup. In practical terms, the pre- and post-processing tend to add relatively minor (or controllable) overheads, but the conversion and embedding costs pose more fundamental problems.

The conversion stage can be problematic for two reasons. First, if the conversion is slow, t_{conv} may be sufficiently large to negate any speedup. However, asymptotically t_{conv} should be polynomial in the problem size n, and, in practice for problems suitable for annealing, t_{conv} seems to be relatively small compared to t_{proc} and thus has negligible impact on the ability to find an absolute speedup.

More importantly though is the fact that the QUBO instance resulting from the conversion may be significantly larger than the original problem instance, and thus it can be too large to solve with current quantum annealers. For example, [31] studies the QUBO formulation of the well-known Broadcast Time Problem obtained through a reduction from Integer Programming. For instances of this problem on graphs with less than 20 vertices the corresponding QUBO formulation required up to 1000 binary variables (and thus logical qubits) which, especially once the problem is embedded in the physical graph, is beyond the reach of current quantum annealing hardware.

The computational cost of embedding the QUBO instance in the hardware graph is, in absolute terms, perhaps even more of an obstruction to successful applications of quantum annealing in its current state. As mentioned earlier, when using standard heuristic algorithms the embedding time t_{embed} is generally (at best) comparable to t_{proc} (and, indeed, T_C) and often much longer. Like the issues associated with the conversion, if sufficiently many qubits are available (i.e., quadratic in the QUBO problem size) and can reliably be annealed, then this embedding can be done quickly and this problem could be neglected. However, this is certainly not the current situation, and ways to mitigate the dominant effect of t_{embed} will be needed if quantum annealing is to be successfully applied in its current state or imminent future.

These difficulties in turning a raw quantum speedup into a practical advantage for practical problems have led to significant interest in "hybrid classical-quantum" approaches (also called "quassical" computations by Allen, see [12]): hopefully, by combining quantum annealing with classical algorithms may allow otherwise inaccessible speedups to be exploited.[8] Several such hybrid approaches have aimed to overcome the resource limitation arising from the fact that practical problems typically require more qubits than are available on existing devices (as a result of the expansion in number of variables during the conversion stage discussed above) [16, 17]. Such proposals instead provide algorithms that utilise quantum annealing on smaller, more manageable subproblems before combining the results classically into a solution for the larger problem at hand. Other hybrid approaches have aimed to combine quantum annealing with classical annealing and optimisation techniques, in particular by using quantum annealing to perform local optimisations and classical techniques to guide the global search direction [18, 19]. These approaches aim to make the most of both quantum advantages (e.g. tunnelling) and classical ones (the ability to read and copy intermediate states).

3.1 Hybrid computing to mitigate minor-embedding costs

Although hybrid approaches have also looked at improving the robustness and quality of embeddings [45], to the best of our knowledge such approaches have not been used to try and mitigate the cost of performing the embedding itself, which, we recall, is often prohibitive to any

[8] We note that hybrid approaches have been also proposed (explicitly and implicitly) in other models of quantum computation too. For example, measurement based computation can be seen a hybrid approach: one starts with a quantum state and performs iterative rounds of quantum measurements and classical computations determining future measurements [43, 44].

speedup. In this paper we propose a general hybrid approach to tackle precisely this problem. In particular we aim to show how a raw speedup that is negated by the embedding time (i.e., in particular when $t_{\text{proc}} < T_C$ but $T_Q > T_C$) can nonetheless be exploited to give a practical speedup to certain computational problems.

Our approach is motivated by another hybrid quantum-classical algorithmic proposal which predates the rise of quantum annealing and was introduced with the aim of exploiting Grover's algorithm—the well-known black-box algorithm for quantum unordered database search [46]—in practical applications [15]. The motivation in this case was the realisation that, although Grover's algorithm offers a provable quantum speedup, it applies in rather artificial scenarios: it assumes the existence of an unsorted quantum database, when generally a more practical database design would allow for even better speedups, and in most conceivable practical scenarios a costly pre-processing step is needed to prepare the database which immediately negates the quantum speedup. The authors showed, however, that some more complex practical problems can be approached by solving a large number of instances of unstructured database searches on a single database—precisely the problem that Grover's algorithm is applicable to. Specifically, they looked at practical problems in computer graphics, such as intersection detection in ray-tracing algorithms.[9] The need to run Grover's algorithm many times to solve such problems means that the cost of preparing and pre-processing the database can be averaged out over all the runs, thus allowing the theoretical quantum speedup to be recovered.

Although the hybrid approach of [15] applies to a very different situation than that of quantum annealing, there are some clear similarities between the prohibitive costs of preparing the database for Grover's algorithm, and that of performing the embedding prior to annealing. It is thus reasonable to adopt a similar approach of looking at more complex algorithms that require solving sets of related problems on a quantum annealer might allow a quantum speedup to be obtained where it was previously hidden behind the cost of the embedding. In particular, it might be easier to observe (and thus take advantage of) a quantum speedup by looking at algorithms that require a large number of calls to a quantum annealer as a subroutine, rather than trying to observe a speedup for solving an individual problem instance on an annealer (e.g., a single instance of an NP-complete problem such as the Independent Set problem via a reduction to a single QUBO instance).

The crucial condition for such a problem to be amenable to this hybrid approach is that *the repeated calls to the quantum annealer should be made with the same logical graph embedding*, or *permit an efficient method to construct the embedding for one call from the previous ones.* If this condition is satisfied, the cost of the embedding, t_{embed}, can thus be spread out over the several calls, allowing a raw quantum speedup to be exploited. There are several conceivable ways such a scenario could naturally occur in realistic algorithmic problems, and we will discuss and analyse an example in detail in the following sections. Perhaps the most trivial would be that where all (or most) solutions to a highly-degenerate problem are required to be found, rather than simply a single one. Although such a scenario is clearly very suitable for quantum annealing, given its intrinsic ability to randomly sample solutions, there are other, perhaps more subtle, situations where this hybrid approach could be applied. For example, one may need to solve a large number of instances of a problem, $P_1, \ldots, P_m$, where the instances P_i differ in some parameters, but where the embedding is independent of these parameters (e.g., if they are encoded in the weights rather than couplings of the logical graph), or if the logical graphs G_i of each instance P_i differ only slightly and are all subgraphs of a single logical graph G that can be embedded.[10] These examples are certainly not definitive, and other situations suitable for this

[9] Here, one must determine the intersections between large numbers of *a priori* unordered three-dimensional objects, which can be rephrased as a search for an initially unknown number of items in an unordered database.

[10] Of course, one would want G to be not much larger than the G_i, otherwise the embedding of G is unlikely to allow one to compute *good* embeddings of the G_i.

hybrid approach are bound to be uncovered.

In order to see how this hybrid approach can help exploit a quantum speedup, we will consider the following general description of a quantum annealing algorithm based on the hybrid approach described above (a more precise analysis would necessarily depend in part on the algorithm in question): some initial classical processing is performed, the embedding of a logical graph into the physical graph is computed, m instances of a QUBO problem are solved on a quantum annealer, with some classical pre- and post-processing occurring between instances, and some final classical computation is optionally performed.

More formally, let us call the overall problem the hybrid algorithm solves R, and the m problem instances that must be solved to do so, $P_1, \ldots, P_m$. Recall that the time to solve a single instance P_i on an annealer is $T_Q(P_i)$; as we noted earlier this is, in practical situations, generally dominated by the cost of the embedding and the quantum processing, so $T_Q(P_i)$ can be approximated, for simplicity, as

$$\begin{aligned} T_Q(P_i) &= t_{\text{conv}}(P_i) + t_{\text{embed}}(P_i) + t_{\text{pre}}(P_i) + t_{\text{proc}}(P_i) + t_{\text{post}}(P_i) \\ &\approx t_{\text{embed}}(P_i) + t_{\text{proc}}(P_i), \end{aligned}$$

where we have explicitly included the dependence on the problem instance. The hybrid algorithm will thus take time

$$\begin{aligned} T_H(R) &\approx t_1(R) + t_{\text{embed}}(P_1) + \sum_i \big(t_{\text{proc}}(P_i) + t_2(P_i)\big) \\ &\approx t_1(R) + t_{\text{embed}}(P_1) + \sum_i t_{\text{proc}}(P_i), \end{aligned} \tag{4}$$

where $t_1(R)$ encapsulates any initial and final classical processing associated with combining the solutions P_i, and $t_2(P_i)$ is the classical calculation associated with each iteration, which we have assumed to be small compared to $t_{\text{proc}}(P_i)$ since this should simply encompass minor pre- and post-processing between annealing runs, and thus be negligible if the problem is amenable to the hybrid approach.[11] Note that we have made use of the assumption that $t_{\text{embed}}(P_1) \approx t_{\text{emded}}(P_i)$ for $i > 1$, which is a criterion in the suitability of a problem for this hybrid approach.

We note immediately that a standard approach with a quantum annealer, performing the embedding for each instance P_i, would take time

$$T_{\text{std}}(R) \approx t_1(R) + \sum_i \big(t_{\text{embed}}(P_i) + t_{\text{proc}}(P_i)\big).$$

In practice, one could envisage exploiting classical parallelism to reduce the cost of performing the embedding m times by a constant factor. For simplicity, we will assume that such parallelism is not used, and as long as m is large enough the same conclusions hold. Thus, since in practice t_{embed} is comparable to, if not larger, than t_{proc}, we already have

$$T_H(R) \ll T_{\text{std}}(R).$$

Although this conclusion may seem somewhat trivial, it is important in that it shows already how annealing can provide much larger practical gains for such complex algorithmic problems.

More importantly, *it may allow a raw quantum speedup to be exploited practically.* To see this, let us consider the case when the best classical algorithm can solve a single instance P_i in time

[11] More precisely, one expects the annealing time to be exponential in general, and if an exponential amount of classical processing is also required, it seems likely that no speedup will be possible. This condition could nonetheless be relaxed to obtain an advantage with the hybrid approach, as long as a raw speedup is still present when the annealing and processing times are combined (i.e., $t_{\text{proc}} + t_2$), but negated by the embedding if the annealer is used in the standard, more naive, way; however, we make this assumption to simplify our analysis.

$T_C(P_i)$.[12] We are interested, in particular, in the case when a raw quantum speedup (i.e., $t_{\text{proc}}(P_i) < T_C(P_i)$) is negated by the embedding (i.e., $T_Q(P_i) > T_C(P_i)$). Although the standard classical approach to solving R is to use the classical algorithm to solve each P_i, and would thus take time $t_1(R) + \sum_i T_C(P_i)$, we should not assume this is the best classical approach to solving R, and for a fair comparison the hybrid approach should be benchmarked against the best known classical algorithm for R.

It is, of course, possible that, for certain problems, a much more efficient classical algorithm exists for solving R when m is large enough (e.g., there might be an efficient way to map solutions of P_i to P_j). Such problems are thus not suitable for such a hybrid approach, and so are not of particular interest to us. Nonetheless, in general a classical algorithm for R may be more intelligent than the standard approach as certain, necessarily minor,[13] parts of the computation are likely to be common to solving several P_i. Specifically, we can thus rewrite $T_C(P_i) = t_3(P_i) + t_4(P_i)$, where t_3 is small compared to t_4. The best classical algorithm can then, rather generally, be considered to take time

$$T_C^{\text{best}}(R) = t_5(R) + t_3(P_1) + \sum_i t_4(P_i) = t_6(R) + \sum_i t_4(P_i),$$

where $t_6(R) = t_5(R) + t_3(P_1)$ and $t_5(R)$ encapsulates any additional global processing (in analogy to $t_1(R)$ for the quantum approaches). Crucially, unless the raw quantum speedup is small, we will also have $t_{\text{proc}}(P_i) < t_4(P_i)$.

It is thus easy to see that,

for large enough m (i.e., number of P_i to be solved), we have $T_H(R) < T_C^{best}(R)$,

and thus the raw quantum speedup will translate into an absolute speedup for the hybrid algorithm. The precise value of m for which such a speedup is obtained will, of course, depend on the problem instances themselves, since the runtime can in practice depend heavily on this. Moreover, although m depends on the problem R (it may, for example, scale with the problem size, or be fixed), this analysis shows that there are problems for which this hybrid approach can turn a raw quantum speedup into a practical one.

It is important to reiterate that the quantum (and, if applicable, classical) times should be calculated using the TTS metric for each problem instance in order to correctly take into account the probabilistic nature of the quantum (and, potentially, classical) algorithms, just as when benchmarking the performance of an annealer on individual problem instances. The performance of the overall hybrid algorithm is thus itself probabilistic and assessed in a similar fashion.

Finally, we reiterate that such a hybrid approach can, of course, only provide a quantum speedup if a raw quantum speedup exists. The existence of such speedups for practical problems remains heavily debated, but the purpose of the hybrid approach is to exploit such an advantage when or if it is present.

4 Case study: Dynamically weighted maximum-weight independent set

To illustrate the proposed hybrid approach, we discuss in detail a concrete example both from a theoretical and experimental viewpoint. We first present the problem, which is intended

[12] We emphasise that, since we are interested in practical, not only asymptotic, gains, we can not easily assume that $T_C(P_i) = T_C(P_j)$ for $i \neq j$.

[13] If not, then again the problem is not suitable for the hybrid approach, as a much more efficient classical algorithmic approach exists.

as a proof-of-concept example rather than one of any particular practical application, before discussing an experimental implementation on a D-Wave quantum annealer and analysing the results of this experiment.

Our problem is based on a variant of the well-known independent set problem, the maximum-weight independent set (MWIS) problem. More precisely, we consider the question of solving many instances of this problem with different (dynamically assigned) weights on the same graph.

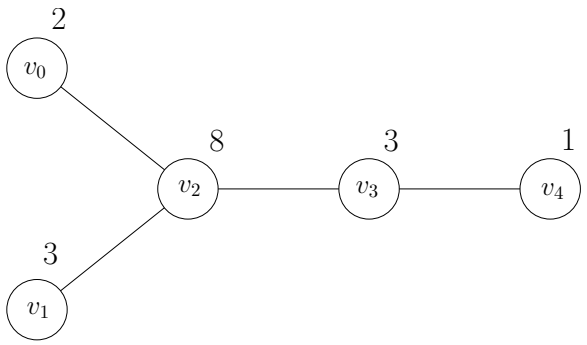
4.1 Maximum-weight independent set

Recall that an *independent set* V' of vertices of a graph $G = (V, E)$ is a set $V' \subseteq V$ such that for all $\{u, v\} \in E$ we have $\{u, v\} \not\subseteq V'$.

Maximum-Weight Independent Set (MWIS) Problem:

Input: A graph $G = (V, E)$ with positive vertex weights $w : V \to \mathbb{R}^+$.
Task: Find an independent set $V' \subseteq V$ such that maximises $\sum_{v \in V'} w(v)$ over all independent sets of G.

Note that the number of vertices in a maximum weighted independent set may be of smaller size then the number for its maximum independent set. For example, consider the weighted graph shown in Figure 2(a). The vertices $\{v_2, v_4\}$ have total weight 9, while the larger set $\{v_0, v_1, v_3\}$ has only total weight 8.



(a)

	v_0	v_1	v_2	v_3	v_4
v_0	-2	0	12	0	0
v_1	0	-3	12	0	0
v_2	0	0	-8	12	0
v_3	0	0	0	-3	12
v_4	0	0	0	0	-1

(b)

Figure 2: An example of (a) a vertex-weighted graph and (b) its MWIS QUBO matrix (cf. Section 4.3).

The general MWIS problem is NP-hard since it encompasses, by restriction, the well-studied non-weighted version [47]. One should note, however, that for graphs of bounded tree-width, the MWIS problem is polynomial-time solvable using standard dynamic programming techniques (see [48]).

We finish the presentation of the MWIS problem by mentioning an important application of it that was studied in [49, 50]. Hence, although the example we presented is intended simply as a proof-of-concept, it is not far removed from computational problems of interest. Suppose we have a wireless network consisting of several nodes and each node has a certain amount of data it needs to transfer. The problem consists in finding the set of nodes that should be given permission to transfer so that the total amount of data output is maximised under the condition that none of the transmissions can interfere with each other. If the vertices of the graph $G = (V, E)$ are devices in the network, the weight associated with each node represents the amount of data it needs to transfer and each edge in E codes the potential interference between its two endpoints

(so that only one of them can be transferring at a given time), then finding the optimal schedule for transmission is equivalent to finding the maximum-weight independent set of G.

4.2 Dynamically weighted MWIS

Although the MWIS can be readily transformed into a QUBO problem (as we show below), by itself it is not directly suitable for the hybrid approach we proposed. However, a simple variation that we propose here is indeed suitable.

Consider the network scheduling problem presented in the previous subsection. Suppose that each node in the network now has multiple messages it needs to send with various sizes, but the underlying structure of the graph remains the same (i.e., the same set of devices with unchanged potential interference), but the weight associated with each node will now change over time. Finding the optimal transmission schedule over time in this network is the same as finding the maximum weighted independent set of the graph with multiple weight functions.

Formally, we have the following problem:

Dynamically Weighted Maximum-Weight Independent Set (DWMWIS) Problem:

Input: A graph $G = (V, E)$ with a set of weight functions $W = \{w_1, w_2, \ldots, w_m\}$ where $w_i : V \to \mathbb{R}^+$ for $1 \leq i \leq m$.
Task: Find independent sets $V_i \subseteq V$ that maximise $\sum_{v \in V_i} w_i(v)$ for each $1 \leq i \leq m$.

This problem is to solve the MWIS problem on G for each of the m weight assignments $w_i \in W$.

For $m = 1$ we obtain again the MWIS problem, but for larger m the problem is suitable for our hybrid approach.

4.3 Quantum solution

We now provide a QUBO formulation for the MWIS Problem. Fix an input graph $G = (V, E)$ with positive vertex weights $w : V \to \mathbb{R}^+$. Let $W = \max\{w(v) \mid v \in V\}$ and let $S > W$ be a "penalty weight". We build a QUBO matrix of dimension $n = |V|$ such that:

$$Q_{(i,j)} = \begin{cases} 0, & \text{if } i > j \text{ or } \{i,j\} \notin E, \\ -w(v_i), & \text{if } i = j, \\ S, & \text{if } i < j \text{ and } \{i,j\} \in E. \end{cases} \tag{5}$$

Theorem 1. *The QUBO formulation given in (5) solves the MWIS Problem.*

Proof. Let $\mathbf{x}$ be a Boolean vector corresponding to an optimal solution to the QUBO formulation (5). Let $D(\mathbf{x}) = \{v_i \mid x_i = 1\}$ be the vertices selected by $\mathbf{x}$.

If $D(\mathbf{x})$ is an independent set then $-x^* = -\mathbf{x}^T Q \mathbf{x}$ is its weighted sum. For two different solutions $\mathbf{x}_1$ and $\mathbf{x}_2$, which correspond to independent sets, the smallest value of $\mathbf{x}_1^T Q \mathbf{x}_1$ and $\mathbf{x}_2^T Q \mathbf{x}_2$ is better.

Now assume $D(\mathbf{x})$ is not an independent set. We will show that the objective function corresponding to $D(\mathbf{x})$ can be improved. Indeed, since $D(\mathbf{x})$ is not independent there must be two vertices v_i and v_j in $D(\mathbf{x})$ such that $\{v_i, v_j\}$ is an edge in the graph. Let $\mathbf{x}_1 = \mathbf{x}$ but set $x_i = 0$, i.e. $D(\mathbf{x}_1) = D(\mathbf{x}) \setminus \{i\}$. We have $\mathbf{x}_1^T Q \mathbf{x}_1 < \mathbf{x}^T Q \mathbf{x} - W + w(v_i) \leq \mathbf{x}^T Q \mathbf{x}$. (Note the second inequality is saturated if and only if v_i is a pendant vertex attached to v_j.) We can repeat this process on improving $\mathbf{x}$ to $\mathbf{x}_1$ until we get an independent set. Thus the optimal value of the

QUBO holds for some independent set. By the conclusion of the second paragraph of this proof, we know that a maximum weighted independent set corresponds to x^*. □

In Figure 2(b) we give the QUBO matrix for the example in Figure 2(a) with penalty entries $P = 12 > W = 8$, [31, 51]. It is easy to see that with $\mathbf{x} = (0, 0, 1, 0, 1)$ we have the minimum value $x^* = \mathbf{x}^T Q \mathbf{x} = -9$. The maximum total weight is thus indeed $-x^* = 9$, as expected.

As a sanity check of the practicality of this solution on real quantum annealing machines, we implemented it on a D-Wave 2X device. For this example it is easy to see that the graph in Figure 2(a) is a subgraph of $K_{4,4}$, hence a trivial embedding is possible.[14] The algorithm gave the expected optimal answer of $\{v_2, v_4\}$ approximately two-thirds of the time, and the non-optimal answer of $\{v_0, v_1, v_3\}$, a third of the time; occasionally other results, such as $\{v_2\}$ or $\{v_0, v_1, v_4\}$ were obtained, although such occasional incorrect solutions are not unexpected for quantum annealers. Further details of the implementation are given in Appendices B and C.

In order to adapt the MWIS solution above to the DWMWIS problem, note that the non-zero entries of the QUBO formulation (5) depend only on the structure of the graph and not on the weight function w. Thus, in order to solve the DWMWIS problem, for each weight assignment w_i the same embedding of the graph into the D-Wave physical graph can be used, meaning that a hybrid algorithm based around the MWIS solution above can readily be implemented (see Appendix D).

More specifically, following the hybrid algorithm described in Section 3.1 for instances $P_1, \ldots, P_m$ (where each P_i uses weight function w_i), we perform the embedding once (entailing a time $t_{\text{embed}}(P_1)$) and then solve the MWIS problem for each weight assignment w_i (taking times $t_{\text{proc}}(P_i)$) using the QUBO solution outlined above. Note that the iteration times $t_2(P_i)$, $1 \leq i \leq m$, in Eq. (4) thus correspond to the time to read in and alter the coupling weights in the QUBO matrix.

4.4 Classical baseline

The main objective of studying the DWMWIS example in detail is to exhibit experimentally the advantage that the hybrid approach can provide over a standard annealing-based approach. Nonetheless, it is helpful to further compare this to the performance of a classical baseline algorithm for comparison and to help highlight this advantage, even if we do not necessarily expect to see an absolute quantum speedup from the hybrid algorithm.

As we discussed in detail in Section 2.4.2, one should ideally compare the hybrid algorithm against the best available classical algorithm for the same problem. However, since our primary concern is not to show an absolute quantum speedup, and studying more closely the performance of various classical algorithms for the DWMWIS problem is somewhat beyond the scope of the present article, we will use a more generic classical algorithm based on a Binary Integer Programming (BIP) formulation of the MWIS problem for illustrative purposes.

To this end, for a given input graph $G = (V, E)$ with positive vertex weights $w : V \to \mathbb{R}^+$, we construct a BIP instance with $n = |V|$ binary variables as follows. To each vertex v_i in G we associate the binary variable x_i, and for notational simplicity we will denote the collection of variables x_i by a binary vector $\mathbf{x} = (x_0, x_1, \cdots, x_{n-1})$. We thus have the BIP problem instance:

$$\begin{aligned} &\text{maximise} \sum_{v_i \in V} w(v_i) x_i \\ &\text{subject to } x_i + x_j \leq 1 \text{ for all } \{v_i, v_j\} \in E. \end{aligned} \tag{6}$$

[14] We took, for example, the embedding $[v_0 \to 0, v_1 \to 1, v_2 \to 4, v_3 \to 2, v_4 \to 7]$ into the first bipartite block of the Chimera graph shown in Figure 1.

Each constraint in (6) enforces the property that no adjacent vertices are chosen in the independent set while the objective function ensures an independent set with maximum sum value is chosen. Assuming we have the binary vector $\mathbf{x}$ which yields the optimal value of objective function (6), we take $D(\mathbf{x}) = \{v_i \mid x_i = 1\}$ to be the set of vertices selected as the maximum weighted independent set.

Theorem 2. *The BIP formulation given in (6) solves the MWIS problem.*

Proof. First, we show that $D(\mathbf{x})$ is an independent set if and only if all the constraints in (6) are satisfied. This is indeed the case, if all the constraints are satisfied, it ensures that for each $\{v_i, v_j\}$ in E, at most one of them is in $D(\mathbf{x})$ by its definition. On the other hand, if any one of the constraint is not satisfied, then it means v_i and v_j are both chosen. Thus $D(\mathbf{x})$ is not an independent set.

Now, let $\mathbf{x}$ be a binary vector corresponding to an optimal solution of BIP formulation (6). Let $D(\mathbf{x}) = \{v_i \mid x_i = 1\}$ be the vertices selected by $\mathbf{x}$. Since $\mathbf{x}$ is the optimal solution, we already have all the constraints of (6) satisfied and and $D(\mathbf{x})$ is therefore a valid independent set. The objective function will ensure that the selected independent set has the maximum value sum. □

The classical baseline[15] we use in the analysis presented in the remainder of this section is based on an implementation of the BIP formulation in Sage Math [52], which has a well developed and optimised Mixed Integer Programming library. To ensure that a fair comparison with the hybrid algorithm is possible, we formulate the classical algorithm for the overall DWMWIS problem such that *the set of constraints in the BIP formulation is only computed once* (cf. the discussion in Section 3.1). This is possible since (in analogy with the need to only perform the embedding once in the quantum solution) the changing weights do not change the constraints of the BIP formulation, and we make use of this to reuse parts of the computation where possible.

4.5 Experimental definition and procedure

To study experimentally the performance of the hybrid DWMIWS algorithm, we compare the performance of three algorithms on a selection DWMWIS problem instances: the "standard" quantum algorithm, in which the embedding is re-performed for each weight assignment; the hybrid DWMWIS algorithm; and the classical BIP-based solution described above.

To this end we analyse the algorithms on a range of different graphs, in particular choosing 155 graphs from a variety of common graph families with between 2 and 126 vertices. The full list of graphs and some of their basic properties (order, size) can be found in the summary of results in Appendix A. Each graph was used to generate a single DWMWIS problem instance with $m = 100$ weight assignments, each randomly generated as floating point numbers rounded to 2 decimal places within the range $[0.0, 1.0)$ using the default pseudo-random generator in Python. Although the choice of m of weight assignments is somewhat arbitrary, our choice was made by the need to balance the ability to solve sufficiently large problems to be able to negate the embedding time against the limited access we had to the quantum annealer. The problem instances were generated as standard adjacency list representations using SageMath [52] with random weights assigned as per the details in Appendix F.

The hybrid DWMWIS algorithm outlined in Section 4.3 was implemented on a D-Wave 2X quantum annealer with 1098 active physical qubits [22]; see Appendix D for the code. The same

[15] Our local linux machine, running Fedora 25 OS, consisted of an Intel Haswell i7 4.0GHz (overclocked to 4.5GHz) with 32GB DDR3 2400Mhz RAM.

procedure is used for the "standard" quantum algorithm, except the cost of the embedding is incurred for each weight assignment (as per Section 3.1).

Since we are primarily interested in negating the impact of the embedding process in general applications, we made use of D-Wave's heuristic embedding algorithm [53] to embed each logical graph in the physical graph. While specialised embedding algorithms may be more effective in certain scenarios, the overall hybrid approach would still be applicable, and by adopting a generic algorithm our results have wider relevance. Each graph was embedded 10 times to estimate t_{embed} for each problem instance. Unfortunately, due to the large number of samples often required to be run for each problem and restrictions on access to the annealer, we were unable to perform a full analysis with each embedding (recall the embedding is non-deterministic). This introduces a potential systematic error since the embedding generally affects the solution quality to some degree; we will discuss this further in the analysis that follows.

Operational parameters for the D-Wave 2X device were determined via an initial testing round (see [54, 55] for further information on D-Wave timing parameters). In line with previous research [8, 25, 39, 40] (cf. Section 2.4.1) we found the minimal annealing time of 20μs to be optimal for all the graphs considered. The programming thermalisation time, which specifies how long the quantum processor is allowed to relax thermally after being programmed with a QUBO problem instance, was chosen as its default value of 1000μs, as this was seen to produce satisfactory results. Between anneals, the processor must similarly be allowed to thermalise, and the default 50μs delay was used. Reading out the result of each anneal takes 309μs on the D-Wave 2X device, so this readout time (and to a lesser extent the thermalisation) dominated the actual annealing time. With minor additional low level processing taken into account, each annealing "sample" has a fixed time of 380.2μs. Although the actual annealing time of 20μs was a minor part of each annealing cycle, this is likely to change in the future as larger problems necessitating longer annealing times become accessible. Moreover, future generations of the machine could have shorter relaxation periods and faster readout times (at least relative to the annealing time, if not in absolute terms) as the physical engineering of the processor is better developed [25, 56].

Finally, our tests were run with D-Wave's post-processing optimisation enabled. While this adds a small overhead in time, this is well within the spirit of hybrid quantum-classical computing, and allowed us to solve more problems. This post-processing method processes small batches of samples while the next batch is being processed [57]. This ensures that it only contributes a constant overhead in time for each MWIS problem instance *independent of the number of samples (and thus of k_{99})*.

To estimate the TTS times T_H and T_{std} described in Section 3.1, one must first estimate k_{99}, as defined in Eq. (2), for each weight assignment w_i. This is done by estimating the probability of success s_i for each such case as $N_{\text{opt}}/N_{\text{total}}$, where N_{total} is the number of annealing cycles performed, while N_{opt} denotes the number of times an optimal solution was found. To determine this ratio accurately for each weight assignment, each problem instance was initially run twice with 1000 samples. Problem instances for which an optimal solution was not found several times for every weight assignment were run a further 5 times; the hardest instances were eventually run a further two times with 2000 samples per run and, for one difficult graph (the bipartite complete graph $K_{12,12}$) a further 14 runs of 2000 samples. By performing many runs (and since each weight assignment is considered separately), random noise due primarily to analogue programming accuracy is largely reduced, and k_{99} is estimated more accurately.

Some problem instances remained unsolved after these runs (i.e., there was at least one weight assignment w_i for which an optimal solution was never found so that k_{99} was undefined) and such problem instances had to be abandoned. As a result, the initial 155 graphs were reduced to 124 for which a running time could be computed and analysed. The fact that such cases were

not uncommon despite the relatively modest size of the graphs highlights limitations of the current state of quantum annealing on more traditional (and, potentially, practical) computational problems.

4.6 Results and analysis

For each DWMWIS problem instance (i.e., for each graph G) the times T_H and T_{std} were calculated, following the approach described in Section 3.1, as

$$T_H = t_{\text{embed}} + \sum_i \left(t_{\text{prog}}(P_i) + k_{99}(P_i) t_{\text{anneal}} + t_{\text{post}}(P_i) \right)$$

and

$$T_{\text{std}} = \sum_i \left(t_{\text{embed}} + t_{\text{prog}}(P_i) + k_{99}(P_i) t_{\text{anneal}} + t_{\text{post}}(P_i) \right),$$

where $k_{99}(P_i)$ is the k_{99} value for weight assignment w_i and $t_{\text{anneal}} = 309\mu$s. As noted in Section 3.1, T_{std} may be reduced by a small constant factor by exploiting classical parallelism, so T_{std} as defined here constitutes an upper bound on the time of a traditional quantum annealing approach. Both $t_{\text{prog}}(P_i)$ and $t_{\text{post}}(P_i)$ are of the order of 20ms (although the latter varies by an order of magnitude more than in the former over different problem instances and runs). Note that the processing time t_{proc} defined earlier is, for this approach to the DWMWIS problem, given by

$$t_{\text{proc}} = t_{\text{prog}}(P_i) + k_{99}(P_i) t_{\text{anneal}} + t_{\text{post}}(P_i).$$

The classical time T_C was taken as the processor time for the classical algorithm described earlier.

A detailed summary of the overall times for each graph is given in Appendix A. These results are summarised in Figures 3(a) and 3(b), which show how the hybrid times T_H compare to both T_{std} and T_C. Error bars are calculated from the observed variation in t_{embed}, the number of optimal solutions found N_{opt}, and the post-processing time t_{post}. Of these, the error in t_{post} is the dominant factor, and largely arises from the uncontrollability of the post-processing environment, which is performed remotely within the D-Wave processing pipeline. However, this variation did not result in any significant variation in success probability of the annealing, so it seems the computational effort expended on post-processing was nonetheless constant. Indeed, we note that in some earlier runs the post-processing was performed 20 times faster with no noticeable change in the quality of solution. Given that post-processing contributes non-negligibly to T_H and T_{std}, this could significantly effect the overall times. We discarded these results to present a conservative analysis and the overall conclusions are not affected by this, but we note that, with increased control of the classical post-processing, the quantum times could be significantly reduced.

As noted in the previous section, practical and logistical constraints prevented us from taking the variation due to different embeddings of each graph fully into account. To assess the possible magnitude of this effect, we tested one relatively difficult graph (Shrikhande) and found that consideration of the embedding roughly tripled the error in T_H, changing the value from $12,800^{+370}_{-240}\mu$s to $15,300 \pm 1,280\mu$s. While this variation would thus generally be a significant source of error, the variation it induces will not be large enough to affect any of our conclusions significantly, even if the inability to take this into account is admittedly regrettable.

First and foremost, from the results shown in Figure 3(a) the extent of the advantage of the hybrid approach is evident. Indeed, this is to be expected given that, for a given DWMWIS problem, they differ (by definition) by $99 \times t_{\text{embed}}$. Although this might seem a trivial confirmation of this fact, the results help illustrate the extent of the advantage that the hybrid approach can

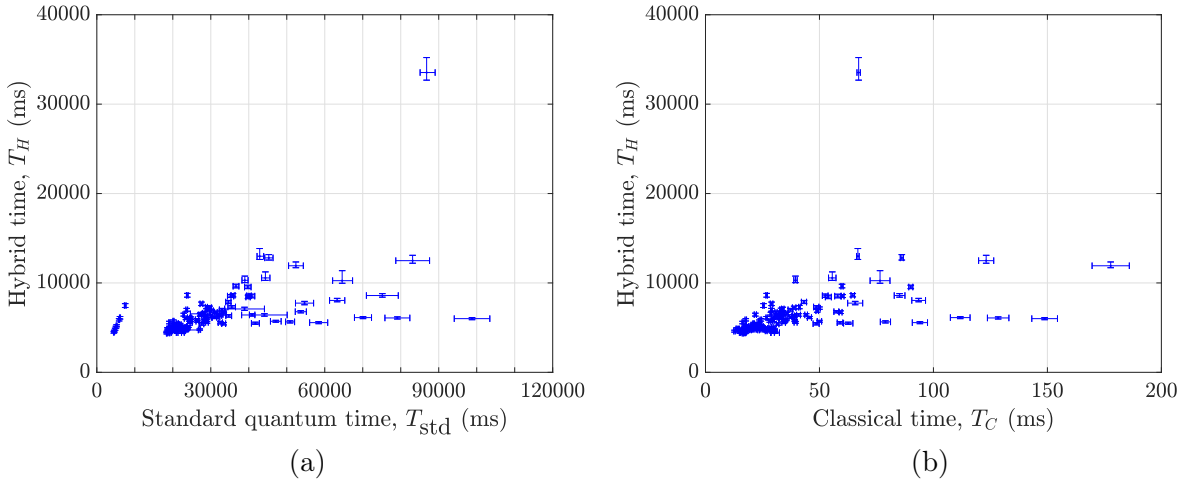


Figure 3: Plots of (a) an upper bound for T_{std} against T_H; and (b) T_C against T_H for each DWMWIS problem instance. All times are in ms.

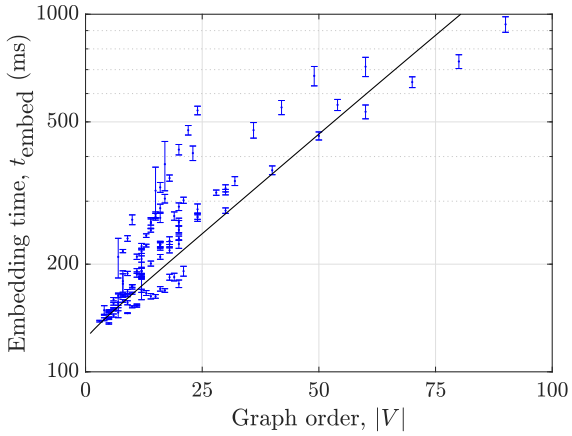


Figure 4: Plot of graph order $|V|$ against the embedding time t_{embed} on a logarithmic time scale.

have for such problems, a consequence of the absolute cost of the embedding. This is visible in Figure 4, showing t_{embed} as a function of the number of vertices in a graph. Although there is a large variation in the embedding times (since, naturally, some graph families are easier to embed than others), a nonlinear regression analysis shows that the dependence on graph order is most consistent with an exponential scaling, as expected. Moreover, from the figure one sees that, even for these relatively small graphs, t_{embed} quickly approaches 1s.

From Figure 3(b) it is also evident that no absolute quantum speedup was observed using the hybrid algorithm, and indeed there is a vast difference in scale between T_C and T_H: the "hardest" problem was solved classically in less than 200ms, whereas the hybrid algorithm required almost 60 times as much time to solve it correctly. The inability to observe any raw speedup is hardly surprising when one notes that, even if $k_{99} = 1$ and $t_{\text{embed}} = t_{\text{post}} = 0$, the fact that $t_{\text{prog}} \approx 20$ms means that that one would have $T_H > 2000$ms. The programming time thus adds an essentially constant overhead, which would have less of an impact as larger problems (for which k_{99} is much larger) become solvable.

Although no overall raw speedup was observed, the experiment nonetheless illustrated the advantage of the hybrid approach over the standard quantum one which, we recall, was the primary goal. It is nonetheless interesting to examine the scaling behaviour of the hybrid algorithm in

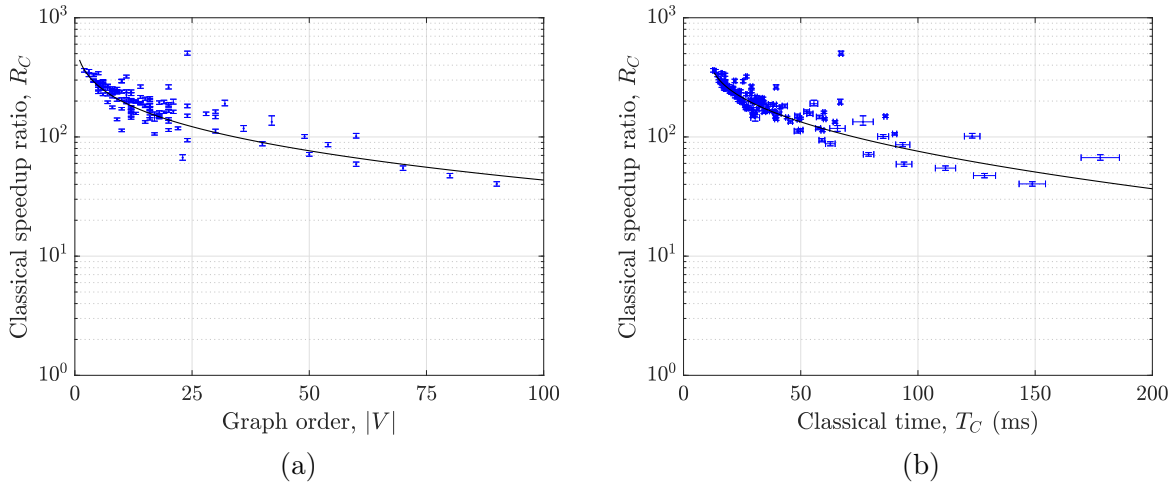


Figure 5: Logarithmic plots of the scaling behaviour of the classical speedup ratio R_C for the DWMWIS problem instances: (a) graph order $|V|$ against R_C; and (b) classical time T_C against R_C.

comparison to the classical one, to see whether there is any tentative evidence that a speedup might be obtainable once the overheads (such as the embedding and programming times) are sufficiently negated. To analyse this more carefully, it will be useful to look at the "classical speedup ratio" $R_C = T_H/T_C$, which provides a clearer measure of any potential speedup: a value of $R_C < 1$ thus indicates an absolute speedup for the hybrid algorithm.[16]

In Figure 5 we show the scaling behaviour of R_C against the graph order $|V|$, which is proportional to the problem size, and the classical time T_C. These two quantities are reasonable proxies of problem difficulty and thus allow the relationship between the performance of the hybrid algorithm and problem difficulty to be investigated. While the scaling of an algorithm is generally studied with respect to problem size, the fact that our examples span a range of graph families, which might all present different scaling behaviour, means that examining the scaling in terms of problem difficulty, as measured by T_C, has empirical merit.

These figures highlight once more the discrepancy between the hybrid and classical times, with the minimum classical speedup observed being $R_C = 40 \pm 2$. Both figures, however, show that R_C decreases with problem size and difficulty, indicating that, for the problem instances tested, the hybrid algorithm exhibited better scaling behaviour than the BIP-based classical algorithm. Both quantum annealing algorithms and the classical baseline we use (due to it being a relatively generic BIP algorithm) are expected to exhibit some form of exponential scaling, even if the precise complexity of the algorithms is *a priori* unknown. A nonlinear regression analysis shows that the scaling behaviour of R_C is indeed, with respect to both $|V|$ and T_C, most consistent with $R_C \propto \exp(k_H \cdot n^{\ell_H})/\exp(k_C \cdot n^{\ell_C})$, for constants k_H, ℓ_H, k_C, ℓ_C, with the hybrid algorithm scaling slower. Due to the large variation in performance over different graph families, however, there is significant uncertainty in the precise form of this scaling. Indeed, one may wish to extrapolate these fits to estimate when one would obtain $R_C = 1$, at which point the hybrid and classical algorithms require the same amount of time. The uncertainty in the scaling behaviour means that any such extrapolation is equally uncertain, with relatively minor changes in the parameters meaning that any estimated point of "hybrid equality" can vary by at least 50% (the uncertainty is particularly large on the upper end of the scale, meaning that such estimates should at best be taken to provide a lower bound). Moreover, one should caution

[16] We could equally look at the hybrid speedup $T_C/T_H = 1/R_C$, but we choose R_C because it is slightly easier to interpret visually.

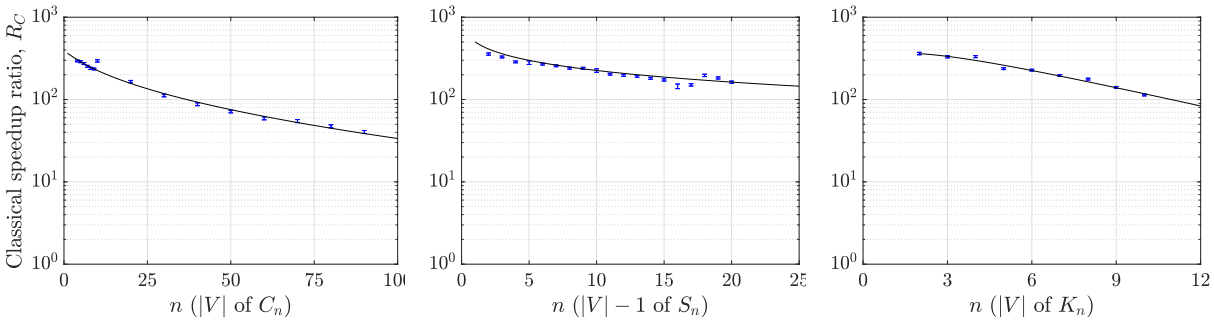


Figure 6: Plots of the classical speedup ratio R_C against n for three families of graphs parameterised by n: (a) the C_n graphs; (b) the S_n graphs; (c) the K_n graphs.

that the scaling may also change for larger problems; indeed, while the minimum annealing time of $t_{\text{anneal}} = 20\mu$s was used for all problem instances here, for larger problems this is no longer likely to be optimal [41, 56]. The consequent need to consider the scaling of t_{anneal} in addition to k_{99} is likely to change future scaling behaviour, as are developments and improvements in future devices (e.g. by decreasing errors arising from noise and limits on the control of qubits). Nonetheless, extrapolation allows a lower bound to be placed on the problem size required for a quantum advantage: we find that such an advantage is not expected until one can *at least* solve graphs of order $|V| = 1,600$ or problems requiring $1,470$ms to solve with the BIP-based classical algorithm.

These numbers are undoubtedly large and some way off what is currently tractable. However, the ability to solve a graph with $1,600$ vertices will depend crucially on the size of the embedding. The worst case of the complete graph K_{1600} would require hundreds of thousands of physical qubits, whereas for other graphs an embedding might be more feasible. It is also worth noting that much of the variance in R_C visible in Figure 5 is due to the data being drawn from several different graph families, and individual graphs that result in outliers. To make more informed estimates, we thus look at the scaling behaviour for different graph families individually. In Figure 6 we show this for the Cycle graphs C_n, Star graphs S_n and the complete graphs K_n (each plotted as a function of n) [52].

Again the scaling behaviour is found to be consistent with a ratio of exponentials, but with much less uncertainty (note that, nonetheless, the log-scale used in Figure 6 makes the uncertainty look smaller that it remains). From these fits, we estimate lower bounds on the point of "hybrid equality" (i.e., when $R_C = 1$) for these three families as being obtained for C_{580}, S_{5618} and K_{38}, respectively. For such families it is possible to give more precise estimates of how many physical qubits would be required to realise such computations. Cycle graphs permit small embeddings, and C_{580} can be embedded in the Chimera graph χ_{10} with 800 physical qubits.[17] As mentioned earlier in Section 2.3, K_{38} can also be embedded in χ_{10}. However, S_{5618} would require a much larger χ_{31} graph.[18] It is thus noteworthy that, at least for certain families of graphs, the prohibitory factor to obtaining a potential quantum speedup is not the number of physical qubits, but the stability and control one has over those qubits. This is pointedly highlighted by noting that many problems that are easily embeddable in D-Wave 2X's physical graph nonetheless fail to be solved by it [14].

As mentioned previously, such estimates as those provided above should only be taken as very

[17] A simple argument shows that there exists at cycle of length at least $\frac{7}{8}|\chi_n|$ by finding a cycle connecting the bipartite blocks, where at least 7 of 8 vertices of each $K_{4,4}$ are spliced into a bigger cycle.

[18] Another argument shows that we can construct in χ_n a spanning caterpillar with $2n^2$ spine vertices with $6n^2$ leaves. Contracting the spine vertices. gives a minor embedding of S_{6n^2}.

conservative lower bounds for when a hybrid speedup may become obtainable: not only may the scaling behaviour change for larger problem instances, but one should also recall that a speedup over a particular classical algorithm—here the BIP-based solver—only proves a potential quantum speedup. For simple families of graphs such as those discussed above, one expects much more efficient classical algorithms to exist. For example, the MWIS of a complete graph is simply $\max_{v \in V} w(v)$ since the only independent sets are singletons. Compared to the generic classical algorithm used, one might approach $R_C = 1$ at the points estimated above should therefore only be taken as general indicators of improved performance of the quantum annealer. Nonetheless our results show that a "potential" quantum speedup remains plausible in the future for the DWMWIS problem, even if it is currently beyond the capabilities of the D-Wave annealer.

While our results failed to find a quantum speedup and produced only tentative evidence that such a speedup might be obtainable in the future for the DWMWIS problem, the experiment was a successful proof-of-concept for the hybrid paradigm we have presented. In particular, the hybrid algorithm we presented provided large absolute gains over the standard quantum approach and showed good scaling behaviour. As larger and more efficient devices become available and more problems of practical interest are studied, it will become clearer if/when a quantum speedup might be obtainable in practise.

5 Conclusion

In this paper, we presented a hybrid quantum-classical paradigm to quantum annealing algorithms. Our paradigm is relevant in particular for devices in which physical qubits have limited connectivity. Thus a problem of interest must be embedded into the graph this connectivity imposes. This problem is a major, but often neglected, hurdle to practical quantum computing. Indeed, not only does the need to find such an embedding often contribute significantly to the overall computational costs, but the quality or size of embedding used can often significantly affect the performance and accuracy of the quantum algorithm itself [30, 45].

The approach we proposed is not only a hybrid paradigm but serves equally as a guide to identifying problems that may be amenable to quantum annealing. In particular, we identify those problems that require solving a large number of related subproblems, each of which can be directed solved via annealing, may permit a hybrid approach. This is obtained by reusing and modifying embeddings for the related subproblems.

To exemplify this approach in an experimental setting, we identified a problem suitable to such a hybrid approach, called the dynamically-weighted maximum weight independent set problem. We experimentally solved a large number of such instances on a D-Wave 2X quantum annealer, and observed the expected advantage of the hybrid algorithm over a more traditional approach in which a known embedding is not reused. We failed to observe a quantum speedup over classical algorithms, although this was not the main goal of the proof-of-concept experiment. This is perhaps unsurprising given that many examples of quantum annealing competing well with classical algorithms are on problems specifically constructed so that embedding is not an issue [6, 8, 39, 41, 42]. We note that another recent experimental study of the (unweighted) maximum independent sets problem conducted on the D-Wave 2000Q machine (the generation following the 2X device we utilised, for which the number of qubits has been doubled), was similarly restricted to graphs with no more than 70 vertices and also failed to observe a speedup [14]; in principle, the weighted version of the problem should be even harder for D-Wave devices because of analogue programming errors and the extra constraints the weights impose. Nonetheless, our hybrid algorithm showed good scaling behaviour, providing tentative evidence that a quantum speedup might be obtainable in the future.

Our hybrid approach, along with its proof-of-principle implementation, sets the groundwork for addressing more complex problems of practical interest. Choosing correctly suitable problems is a major step in finding practical uses for quantum computers in the near term future, and with deft choices, quantum speedups from hybrid approaches might soon be realisable.

Acknowledgements

We thank N. Allen, C. McGeoch, K. Pudenz and S. Reinhardt for fruitful discussions and critical comments. This work has been supported in part by the Quantum Computing Research Initiatives at Lockheed Martin.

References

[1] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, and J. L. O'Brien, Quantum computers, Nature **464**, 45 (2010).

[2] R. Barends, A. Shabani, L. Lamata, J. Kelly, A. Mezzacapo, U. Las Heras, R. Babbush, A. G. Fowler, B. Campbell, Yu Chen, Z. Chen, B. Chiaro, A. Dunsworth, E. Jeffrey, E. Lucero, A. Megrant, J. Y. Mutus, M. Neeley, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T. C. White, E. Solano, H. Neven, and John M. Martinis, Digitized adiabatic quantum computing with a superconducting circuit, Nature **534**, 222 (2016).

[3] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose, Quantum annealing with manufactured spins, Nature **473**, 194 (2011).

[4] S. Boixo, V. N. Smelyanskiy, A. Shabani, S. V. Isakov, M. Dykman, V. S. Denchev, M. H. Amin, A. Yu Smirnov, M. Mehseni, and H. Neven, Computational multiqubit tunnelling in programmable quantum annealers, Nat. Commun. **7**, 10327 (2016).

[5] D-Wave Systems, Inc., The D-Wave 2000Q™ Quantum Computer Technology Overview, (2017).

[6] S. W. Shin, G. Smith, J. A. Smolin, and U. Vazirani, How "quantum" is the D-Wave machine? (2014), arXiv:1401.7087 [quant-ph].

[7] A. Cho, Quantum or not, controversial computer yields no speedup, Science **344**, 1330 (2014).

[8] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, Defining and detecting quantum speedup, Science **345**, 420 (2014).

[9] J. King, S. Yarkoni, M. M. Nevisi, J. P. Hilton, and C. C. McGeoch, Benchmarking a quantum annealing processor with the time-to-target metric, (2015), arXiv:1508.05087 [quant-ph].

[10] A. D. King, T. Lanting, and R. Harris, Performance of a quantum annealer on range-limited constraint satisfaction problems, (2015), arXiv:1502.02098 [quant-ph].

[11] B. Heim, T. F. Rønnow, S. V. Isakov, and M. Troyer, Quantum versus classical annealing of Ising spin glasses, Science **348**, 215 (2015).

[12] C. S. Calude, E. Calude, and M. J. Dinneen, Adiabatic quantum computing challenges, ACM SIGACT News **46**, 40 (2015).

[13] D. Venturelli, D. J. J. Marchand, and G. Rojo, Job shop scheduling solver based on quantum annealing, (2015), arXiv:1506.08479 [quant-ph].

[14] S. Yarkoni, A. Plaat, and T. Bäck, First results solving arbitrarily structured maximum independent set problems using quantum annealing, (2017).

[15] M. Lanzagorta and J. K. Uhlmann, Hybrid quantum-classical computing with applications to computer graphics, in *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05 (ACM, New York, NY, 2005).

[16] T. T. Tran, M. Do, E. G. Rieffel, J. Frank, Z. Wang, B. O'Gorman, D. Venturelli, and J. C. Beck, A hybrid quantum-classical approach to solving scheduling problems, in *Proceedings of the Ninth International Symposium on Combinatorial Search* (AAAI, 2016).

[17] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New J. Phys. **18**, 023023 (2016).

[18] N. Chancellor, Modernizing quantum annealing using local searches, New J. Phys. **19**, 023024 (2017).

[19] T. Graß and M. Lewenstein, Hybrid annealing: Coupling a quantum simulator to a classical computer, Phys. Rev. A **95**, 052309 (2017).

[20] B. Bauer, D. Wecker, A. J. Millis, M. B. Hastings, and M. Troyer, Hybrid quantum-classical approach to correlated materials, Phys. Rev. X **6**, 031045 (2016).

[21] J. Li, X. Yang, X. Peng, and C.-P. Sun, Hybrid quantum-classical approach to quantum optimal control, Phys. Rev. Lett. **118**, 150503 (2017).

[22] D-Wave Systems, Inc., The D-Wave 2X™ quantum computer technology overview, (2016).

[23] E. Farhi, J. Goldstone, S. Gutman, and M. Sipser, Quantum computation by adiabatic evolution, (2000), arXiv:quant-ph/0001106.

[24] A. D. King, E. Hoskinson, T. Lanting, E. Andriyash, and M. H. Amin, Degeneracy, degree, and heavy tails in quantum annealing, Phys. Rev. A **93**, 052320 (2016).

[25] A. D. King and C. C. McGeoch, Algorithm engineering for a quantum annealing platform, (2014), arXiv:1410.2628 [cs.DS].

[26] C. McGeoch, *Adiabatic Quantum Computation and Quantum Annealing. Theory and Practice* (Morgan & Claypool Publishers, 2014).

[27] A. Mizel, D. A. Lidar, and M. Mitchell, Simple proof of equivalence between adiabatic quantum computation and the circuit model, Phys. Rev. Lett. **99**, 070502 (2007).

[28] W. Istrail, Statistical mechanics, three-dimensionality and NP-completeness: I. universality of intractability for the partition function of the ising model across non-planar surfaces, in *STOC '00 Proceedings of the thirty-second annual ACM symposium on Theory of computing* (ACM, 2000) pp. 87–96.

[29] V. Choi, Minor-embedding in adiabatic quantum computation: I. The parameter setting problem, Quantum Inf. Processing **7**, 193 (2008).

[30] W. Lechner, P. Hauke, and P. Zoller, A quantum annealing architecture with all-to-all connectivity from local interactions, Science Advances **1**, e1500838 (2015).

[31] C. S. Calude and M. J. Dinneen, Solving the broadcast time problem using a D-Wave quantum computer, in *Advances in Unconventional Computing*, Emergence, Complexity and Computation, Vol. 22, edited by A. Adamatzky (Springer International, Switzerland, 2016) Chap. 17, pp. 439–453.

[32] R. Saket, A PTAS for the classical Ising spin glass problem on the chimera graph structure, (2013), arXiv:1306.6943 [cs.DS].

[33] V. Choi, Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design, Quantum Inf. Processing **10**, 343 (2011).

[34] F. Barahona, On the computational complexity of Ising spin glass models, J. Phys. A: Math. Gen. **15**, 3241 (1982).

[35] J. Cai, W. G. Macready, and A. Roy, A practical heuristic for finding graph minors, (2014), arXiv:1406.2741 [quant-ph].

[36] K. L. Pudenz, Parameter setting for quantum annealers, in *20th IEEE High Performance Embedded Computing Workshop Proceedings* (2016) arXiv:1611.07552 [quant-ph].

[37] S. Aaronson, BQP and the polynomial hierarchy, in *STOC '10 Proceedings of the forty-second ACM symposium on Theory of computing* (2010) pp. 141–150.

[38] M. Fürer, Solving NP-Complete problems with quantum search, in *LATIN 2008: Theoretical Informatics*, LNCS, Vol. 4957, edited by Eduardo Sany Laber, Claudson Bornstein, Loana Tito Nogueira, and Luerbio Faria (Springer Berlin Heidelberg, 2008) pp. 784–792.

[39] I. Hen, J. Job, T. Albash, T. F. Rønnow, M. Troyer, and D. A. Lidar, Probing for quantum speedup in spin-glass problems with planted solutions, Phys. Rev. A **92**, 042325 (2015).

[40] D. Venturelli, S. Mandrà, S. Knysh, B. O'Gorman, R. Biswas, and V. Smelyanskiy, Quantum optimization of fully connected spin glasses, Phys. Rev. X **5**, 031040 (2015).

[41] S. Boixo, T. F. Rønnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, Evidence for quantum annealing with more than one hundred qubits, Nat. Phys. **10**, 218 (2014).

[42] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven, What is the computational value of finite-range tunneling? Phys. Rev. X **6**, 031015 (2016).

[43] R. Josza, An introduction to measurement based quantum computation, in *Quantum Information Processing: From Theory to Experiment*, edited by D. G. Anghelakis, M. Christandl, A. Ekert, A. Kay, and S. Kulik (IOS Press, Amsterdam, 2006) Chap. 2, pp. 137–158.

[44] H. J. Briegel, D. E. Browne, W. Dür, R. Raussendorf, and M. Van den Nest, Measurement-based quantum computation, Nat. Phys. **5**, 19 (2009).

[45] W. Vinci, T. Albash, G. Paz-Silva, I. Hen, and D. A. Lidar, Quantum annealing correction with minor embedding, Phys. Rev. A **92**, 042310 (2015).

[46] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings, 28th Annual ACM Symposium on the Theory of Computing (STOC)* (1996) pp. 212–219, arXiv:quant-ph/9605043.

[47] M. R. Garey and D. S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979).

[48] D. Marx, Fixed parameter algorithms. Part 2: Treewidth, (2010), open lectures for PhD students in computer science, University of Warsaw, Poland.

[49] K. Jung and D. Shah, Low delay scheduling in wireless network, in *2007 IEEE International Symposium on Information Theory, Nice* (IEEE, 2007) pp. 1396–1400.

[50] S. Sanghavi, D. Shah, and A. S. Willsky, Message passing for maximum weight independent set, IEEE Transactions on Information Theory **55**, 4822 (2009).

[51] E. D. Dahl, Programming with D-Wave: Map coloring problem, D-Wave Systems Whitepaper (2013).

[52] The Sage Developers, Sagemath, the Sage Mathematics Software System (Version 8.0), (2017).

[53] D-Wave Systems, Inc., Programming with QUBOs, Technical Report Release 2.4 09-1002A-C (2017).

[54] D-Wave Systems, Inc., Measuring computation time on D-Wave Systems, Technical Report 09-1107A-E (2018).

[55] D-Wave Systems, Inc., Developer guide for Python, Technical Report Release 2.4 09-1024A-B (2017).

[56] J. King, S. Yarkoni, J. Raymond, I. Ozdan, A. D. King, M. Mohammadi Nevisi, J. P. Hilton, and C. C. McGeoch, Quantum annealing amid local ruggedness and global frustration, (2017), arXiv:1701.04579 [quant-ph].

[57] D-Wave Systems, Inc., Postprocessing methods on D-Wave Systems, Technical Report Release 2.4 09-1105A-B (2016).

[58] D. W. DeTemple, M. J. Dinneen, K. L. McAvaney, and J. M. Robertson, Recent examples in the theory of partition graphs, Discrete Mathematics **113**, 255 (1993).

A Summary of results for MWDWIS instances

All the standard graphs were produced using SageMath [52] and descriptions of them can be found in the corresponding API; the sole exception is the Dinneen Graph, which is described in [58].

Table 1: Table summarising the 124 graphs defining the DWMWIS problem instances and the average times for the hybrid algorithm, the classical BIP-based algorithm, and the standard quantum annealing approach.

Graph $G = (V, E)$	$\|V\|$	$\|E\|$	T_H (ms)	T_C (ms)	T_{std} (ms)
Bidiakis Cube	12	18	4635 ± 102	26.6 ± 0.4	22851 ± 184
First Blanusa Snark	18	27	5799 ± 120	39.2 ± 0.8	28846 ± 591
Second Blanusa Snark	18	27	6280 ± 139	38.9 ± 0.7	28802 ± 405
Brinkmann	21	42	12988^{+861}_{-363}	66.8 ± 0.6	42876^{+1047}_{-698}
Bucky Ball	60	90	12491^{+599}_{-286}	123.1 ± 3.3	83128^{+4462}_{-4431}
Bull	5	5	4379 ± 90	16.4 ± 0.3	18427 ± 99
Butterfly	5	6	4405 ± 91	17.3 ± 0.3	19137 ± 99
C_4	4	4	4441 ± 89	15.1 ± 0.2	19162 ± 447
C_5	5	5	4785 ± 109	16.6 ± 0.3	19209 ± 133
C_6	6	6	4781 ± 103	17.5 ± 0.3	19532 ± 140
C_7	7	7	4785 ± 102	18.9 ± 0.4	20110 ± 176
C_8	8	8	4743 ± 102	19.8 ± 0.4	20375 ± 174
C_9	9	9	4927 ± 107	20.9 ± 0.3	21084 ± 149
C_{10}	10	10	6453 ± 161	21.9 ± 0.5	22877 ± 194
C_{20}	20	20	5788 ± 142	35.0 ± 0.7	28330 ± 640
C_{30}	30	30	5436 ± 135	48.5 ± 1.3	33394 ± 512
C_{40}	40	40	5490 ± 123	62.6 ± 2.0	41743 ± 1043
C_{50}	50	50	5644 ± 123	78.9 ± 2.2	50867 ± 1190
C_{60}	60	60	5560 ± 120	94.1 ± 3.3	58397 ± 2378
C_{70}	70	70	6122 ± 117	111.8 ± 4.3	70066 ± 2245
C_{80}	80	80	6084 ± 123	128.4 ± 4.7	79117 ± 3279
C_{90}	90	90	6006 ± 120	148.8 ± 5.6	98769 ± 4681
Chvatal	12	24	5899^{+124}_{-122}	35.4 ± 0.4	26372^{+439}_{-438}
Clebsch	16	40	8527^{+172}_{-160}	60.2 ± 0.6	35207^{+818}_{-816}
Coxeter	28	42	8424^{+205}_{-181}	53.9 ± 1.3	39807^{+575}_{-567}
Desargues	20	30	6160^{+126}_{-124}	37.3 ± 0.7	30861 ± 672
Diamond	4	5	4783 ± 106	16.0 ± 0.2	19089 ± 111
Dinneen	9	21	6072 ± 126	29.6 ± 0.6	24724 ± 285
Dodecahedral	20	30	6128^{+124}_{-122}	45.6 ± 0.9	31373^{+997}_{-996}
Double Star Snark	30	45	8527^{+214}_{-192}	58.0 ± 1.3	40801^{+773}_{-767}
Durer	12	18	4643 ± 100	30.3 ± 0.3	23076 ± 254
Dyck	32	48	10562^{+673}_{-275}	55.6 ± 1.6	44380^{+1185}_{-1013}
Ellingham Horton 54	54	81	8043^{+232}_{-152}	93.5 ± 3.0	63265^{+2007}_{-1999}
Errera	17	45	9543^{+201}_{-182}	90.1 ± 0.9	39738^{+867}_{-863}
Flower Snark	20	30	5589 ± 105	39.5 ± 0.7	28992 ± 341
Folkman	20	40	10293^{+471}_{-258}	39.5 ± 0.7	38964^{+853}_{-757}
Franklin	12	18	5030 ± 99	25.3 ± 0.4	23127 ± 165
Frucht	12	18	4842 ± 101	29.5 ± 0.5	23791 ± 349
Goldner Harary	11	27	5716^{+132}_{-119}	28.2 ± 0.4	26486^{+381}_{-377}
2×3 Grid	6	7	5073 ± 140	17.6 ± 0.2	19972 ± 162

3×3 Grid	9	12	5336 ± 150	21.1 ± 0.3	21948 ± 258
3×4 Grid	12	17	5122 ± 107	25.0 ± 0.4	24100 ± 447
4×4 Grid	16	24	5409 ± 140	31.7 ± 0.6	27605 ± 551
4×5 Grid	20	31	6999^{+155}_{-153}	37.2 ± 1.0	32956 ± 693
6×6 Grid	36	60	7743^{+195}_{-184}	65.7 ± 3.3	54679^{+2383}_{-2382}
6×7 Grid	42	71	10252^{+1122}_{-287}	76.6 ± 4.4	64583^{+2739}_{-2516}
7×7 Grid	49	84	8591^{+213}_{-183}	85.2 ± 2.3	75158^{+4197}_{-4195}
Grotzsch	11	20	5793 ± 133	29.7 ± 0.3	24741 ± 324
Heawood	14	21	7663^{+197}_{-193}	29.0 ± 0.6	27542^{+380}_{-379}
Herschel	11	18	5871 ± 145	24.3 ± 0.3	24394 ± 349
Hexahedral	8	12	4803 ± 106	20.6 ± 0.3	20920 ± 145
Hoffman	16	32	7010^{+168}_{-167}	33.2 ± 0.6	29453 ± 433
House	5	6	4700 ± 110	16.9 ± 0.3	19292 ± 113
Icosahedral	12	30	7177^{+138}_{-125}	50.0 ± 0.4	29413^{+422}_{-418}
K_2	2	1	4607 ± 109	12.7 ± 0.3	4607 ± 109
K_3	3	3	4821 ± 118	14.6 ± 0.3	4821 ± 118
K_4	4	6	5875 ± 131	17.7 ± 0.3	5875 ± 131
K_5	5	10	5210 ± 119	22.0 ± 0.2	5210 ± 119
K_6	6	15	6101 ± 143	26.8 ± 0.3	6101 ± 143
K_7	7	21	6546^{+158}_{-157}	33.5 ± 0.2	27296 ± 2667
K_8	8	28	7293 ± 180	41.2 ± 0.4	28836 ± 290
K_9	9	36	6883 ± 164	49.0 ± 0.5	30247 ± 457
K_{10}	10	45	6726^{+153}_{-148}	59.3 ± 0.6	33090^{+856}_{-855}
$K_{2,3}$	5	6	5570 ± 142	16.3 ± 0.2	19083 ± 152
$K_{3,3}$	6	9	4486 ± 103	17.7 ± 0.3	4486 ± 103
$K_{3,4}$	7	12	5147 ± 125	19.3 ± 0.3	19641 ± 487
$K_{4,4}$	8	16	5036 ± 123	21.4 ± 0.3	5036 ± 123
$K_{4,5}$	9	20	5729 ± 131	23.6 ± 0.3	20173 ± 136
$K_{5,5}$	10	25	7470 ± 215	25.4 ± 0.3	7469 ± 215
$K_{5,6}$	11	30	8619 ± 212	26.8 ± 0.3	23805 ± 216
$K_{5,7}$	12	35	6563 ± 155	28.7 ± 0.4	28026 ± 292
$K_{5,8}$	13	40	4789 ± 74	30.4 ± 0.4	27103 ± 214
$K_{5,9}$	14	45	6705^{+154}_{-151}	30.9 ± 0.5	31346^{+352}_{-351}
$K_{6,6}$	12	36	6992.0 ± 159	28.9 ± 0.3	23674 ± 231
$K_{6,7}$	13	42	6279.8 ± 125	31.4 ± 0.4	30079 ± 305
$K_{6,8}$	14	48	6353.1 ± 131	33.3 ± 0.5	32331 ± 539
$K_{6,9}$	15	54	7089^{+192}_{-168}	33.9 ± 0.5	38878^{+5248}_{-5247}
$K_{7,7}$	14	49	6480 ± 132	33.5 ± 0.4	32279 ± 941
$K_{7,8}$	15	56	6563 ± 154	35.8 ± 0.5	33432 ± 599
$K_{8,8}$	16	64	6319 ± 150	38.4 ± 0.6	34722 ± 761
$K_{8,9}$	17	72	6416^{+145}_{-137}	40.8 ± 0.8	44115 ± 5996
$K_{9,9}$	18	81	6424 ± 134	44.1 ± 0.6	40895 ± 712
$K_{10,10}$	20	100	5711 ± 109	50.0 ± 1.0	47113 ± 1408
$K_{11,11}$	22	121	6782^{+134}_{-130}	57.4 ± 1.1	53698 ± 1458
$K_{12,12}$	24	144	33536^{+1674}_{-852}	67.2 ± 0.7	86818^{+2241}_{-1717}
Kittell	23	63	11920^{+427}_{-217}	177.8 ± 8.2	52401^{+1959}_{-1924}
Krackhardt Kite	10	18	5048 ± 99	29.3 ± 0.4	22155 ± 263
Markstroem	24	36	5547 ± 130	59.0 ± 1.2	32525 ± 568
McGee	24	36	7309^{+155}_{-148}	48.6 ± 1.2	35504^{+1003}_{-1002}
Moebius Kantor	16	24	6420 ± 155	31.4 ± 0.6	27170 ± 361
Moser Spindle	7	11	5326 ± 131	23.4 ± 0.4	21473 ± 241

Nauru	24	36	7862^{+180}_{-171}	43.2 ± 1.1	34622^{+702}_{-700}
Octahedral	6	12	5461 ± 133	21.1 ± 0.3	21262 ± 219
Pappus	18	27	6618 ± 179	34.1 ± 0.7	28259 ± 398
Petersen	10	15	5069 ± 108	24.5 ± 0.4	22275 ± 183
Poussin	15	39	8621^{+195}_{-182}	64.6 ± 0.8	35846^{+529}_{-525}
Q_3	8	12	5153 ± 99	20.7 ± 0.2	22597 ± 1180
Q_4	16	32	6091 ± 121	33.0 ± 0.6	28643 ± 391
Robertson	19	38	9635^{+220}_{-187}	59.9 ± 0.5	36633^{+764}_{-755}
S_2	3	2	4858 ± 127	13.6 ± 0.2	18580 ± 147
S_3	4	3	4849 ± 105	14.7 ± 0.2	18738 ± 171
S_4	5	4	4506 ± 85	15.7 ± 0.3	18406 ± 93
S_5	6	5	4977 ± 103	17.7 ± 0.8	19204 ± 178
S_6	7	6	4766 ± 102	17.7 ± 0.3	20319 ± 899
S_7	8	7	4819 ± 98	18.8 ± 0.3	22570 ± 1238
S_8	9	8	4807 ± 94	20.0 ± 0.4	20251 ± 225
S_9	10	9	4994 ± 125	20.9 ± 0.3	20042 ± 159
S_{10}	11	10	5290 ± 156	23.5 ± 0.9	20457 ± 222
S_{11}	12	11	4738 ± 92	23.3 ± 0.4	23587 ± 3131
S_{12}	13	12	4814 ± 100	24.4 ± 0.4	21258 ± 281
S_{13}	14	13	4896 ± 98	25.6 ± 0.4	21003 ± 300
S_{14}	15	14	4772 ± 90	26.3 ± 0.6	20860 ± 211
S_{15}	16	15	4738 ± 104	27.3 ± 0.6	21627 ± 270
S_{16}	17	16	4432 ± 84	30.6 ± 1.9	21143 ± 216
S_{17}	18	17	4444 ± 84	29.5 ± 0.8	22650 ± 361
S_{18}	19	18	6113 ± 122	31.1 ± 0.8	24339 ± 471
S_{19}	20	19	6020 ± 123	32.9 ± 0.7	23474 ± 443
S_{20}	21	20	5569 ± 121	34.3 ± 0.8	24497 ± 619
Shrikhande	16	48	12803^{+367}_{-244}	86.1 ± 0.7	45275^{+1106}_{-1072}
Sousselier	16	27	7231^{+171}_{-169}	38.9 ± 0.8	29675^{+531}_{-530}
Thomsen	6	9	5220 ± 137	17.7 ± 0.2	20555 ± 190
Tietze	12	18	4927 ± 113	27.6 ± 0.3	23380 ± 216
TutteCoxeter	30	45	8566^{+191}_{-179}	52.5 ± 1.2	40058^{+594}_{-591}
Wagner	8	12	4817 ± 111	22.0 ± 0.3	21004 ± 191

B Script to run MWIS instances on D-Wave

```
#!/usr/bin/env python
# QUBO (with embedding) -> Ising -> DWave

import sys, time, math, traceback

from dwave_sapi2.remote import RemoteConnection
from dwave_sapi2.util import get_hardware_adjacency
from dwave_sapi2.embedding import embed_problem, unembed_answer
from dwave_sapi2.util import qubo_to_ising
from dwave_sapi2.core import solve_ising

from sys import exc_info

# coupler streingth for embedded qubits of same variable
```

```
#
s,s2=1.0,1.0
if (len(sys.argv)==2): s = float(sys.argv[1])
if (len(sys.argv)==3): s,s2 = float(sys.argv[1]),float(sys.argv[2])
print 'Embed scale=',s,s2
assert 0 <= s <= 1

# read input
#
line=sys.stdin.readline().strip().split()
n=int(line[0])
print('Logical qubits used=', n)

Q = {}
for i in range(n):
    line=sys.stdin.readline().strip().split()
    for j in range(n):
        t = float(line[j])
        if j>=i and t!=0: Q[(i,j)]=t
print 'Q=',Q

(H,J,ising_offset) = qubo_to_ising(Q)
print 'H=',H
print 'J=',J
print 'ising_offset=',ising_offset

# scale by maxV
#
maxH=0.0
if len(H): maxH=max(abs(min(H)),abs(max(H)))
maxJ=max(abs(min(J.values())),abs(max(J.values())))
maxV=max(maxH,maxJ)

for i in range(n):
    if len(H)>i:
        H[i]=s2*H[i]/maxV
    for j in range(n):
        if j>=i and (i,j) in J:
            J[(i,j)]=s2*J[(i,j)]/maxV

embedding=eval(sys.stdin.readline())
print 'embedding=', embedding
print 'Physical qubits used= %s' % sum(len(embed) for embed in embedding)

# create a remote connection using url and token and connect to solver
#
url = "something"
token = "secret"
solver_name = "DW2X"

print('Attempting to connect to network...')
try:
    remote_connection = RemoteConnection(url, token)
    solver = remote_connection.get_solver(solver_name)
except:
    print('Error: %s %s %s' % sys.exc_info()[0:3])
    traceback.print_exc()

A = get_hardware_adjacency(solver)

# Embed problem into hardware
(h0, j0, jc, new_emb) = embed_problem(H, J, embedding, A, \
                      h_range=[-2.0, 2.0], j_range=[-1.0, 1.0] )
```

```
h1= [val*s for val in h0]
j1 = {}
for (key, val) in j0.iteritems():
    j1[key]=val*s
j1.update(jc)
print 'new_emb=',new_emb
print 'h1=',h1
print 'j1=',j1

# call the solver
#
annealT,num=20,1000
print 'annealT=',annealT,'num=',num
result = solve_ising(solver, h1, j1, num_reads=num, annealing_time=annealT)
print 'result:', result

newresult = unembed_answer(result['solutions'], new_emb, \
                           broken_chains='discard', h=H, j=J)
print 'newresult:', newresult

for i, (embsol, sol) in enumerate(zip(result['solutions'], newresult)):
  weight=0
  for j in range(n):
      if sol[j]==1: weight -= Q[(j,j)]
  print 'solution', i, 'size=', sum(x==1 for x in [sol[j] for j in range(n)]), \
        'weight=', weight, [(1+sol[j])/2 for j in range(n)] # Boolean/QUBO
```

C Program to create MWIS QUBO instances

```
#!/usr/bin/env python
# Max Independent Set of graph to QUBO Hamiltonian objective

import sys, networkx as nx

def read_graph(infile=sys.stdin):  # adjacency list format
    n=int(infile.readline().strip())
    G=nx.empty_graph(n,create_using=nx.Graph())
    for u in range(n):
        neighbors=infile.readline().split()
        for v in neighbors: G.add_edge(u,int(v))
    return G

if len(sys.argv)==2:
    gfile=open(sys.argv[1],'r')
    G=read_graph(gfile)
    W=[float(w) for w in gfile.readline().split()]
    gfile.close()
else:
    G=read_graph()
    W=[float(w) for w in sys.stdin.readline().split()]

n=G.order()
S=max(W)+1

print n
for u in range(n):
    for v in range(n):
        if u==v:        print -W[u],
        elif u in G[v]: print S,
        else:           print 0,
```

```
    print
```

D Modifications of MWIS script to run DWMIS instances on D-Wave

```
#!/usr/bin/env python2
# Dynamic MWIS solver

< MWIS preamble script code here >

assert len(sys.argv)==2
weightfile=open(sys.argv[1],'r')

# read and process each weight assignment vector

while True:

    line=weightfile.readline().strip()
    if len(line)==0: break

    W = [round(float(x),2) for x in line.split()]
    print 'W=',W

    for i in range(n):
          Q[(i,i)]=-W[i]

    (H,J,ising_offset) = qubo_to_ising(Q)

    < rest of MWIS body script code here >

weightfile.close()
```

E Sage Script for DMWIS Problem

```
#!/usr/bin/env sage

import sys, networkx as nx
import datetime
def read_graph(graphFile):
  n=int(graphFile.readline().strip())
  G=nx.empty_graph(n,create_using=nx.Graph())
  for u in range(n):
      neighbors=graphFile.readline().split()
      for v in neighbors: G.add_edge(u,int(v))
  return G

total_start_time = datetime.datetime.now()

assert len(sys.argv)==3
graphFile=open(sys.argv[1],'r')

G=read_graph(graphFile)
n=G.order()
graphFile.close()

p=MixedIntegerLinearProgram(solver="GLPK")
x=p.new_variable(binary=True)
```

```
for (u,v) in G.edges():
  p.add_constraint(x[u]+x[v], max=1)

weightFile=open(sys.argv[2],'r')

while True:
  line=weightFile.readline().strip()
  if len(line)==0: break

  W = [round(float(w),2) for w in line.split()]
  print 'W=',W

  p.set_objective(sum(W[j]*x[j] for j in range(n)))

  try:
      solve_start_time = datetime.datetime.now()
      sz=p.solve()
      solve_finish_time = datetime.datetime.now()
  except sage.numerical.mip.MIPSolverException as e:
      pass
  else:
      print p.get_values(x).items()
      print "size", sz
      print "time", (solve_finish_time-solve_start_time)
      print
weightFile.close()

total_end_time = datetime.datetime.now()
print 'Total time is', (total_end_time - total_start_time)
```

F Python Script for Generating Random Weights

```
#!/usr/bin/env python3
# Generate random weights for graphs
# usage: rand_w_gen.py n [cnt]

import sys,random

n = int(sys.argv[1]) # number of vertices
cnt = 1
if len(sys.argv) > 2:
   cnt = int(sys.argv[2]) # number of cases

for x in range(cnt):
    for y in range(n-1): # weights
        print(random.random(), end=' ')
    print(random.random())
```

G Sample output of Appendix B applied to example of Figure 2

```
Embed scale= 1.0 1.0
'Logical qubits used=', 5
'Q=', {(1, 2): 12.0, (0, 0): -2.0, (3, 3): -3.0, (4, 4): -1.0, (1, 1): -3.0,
(2, 3): 12.0, (2, 2): -8.0, (3, 4): 12.0, (0, 2): 12.0}
'H=', [2.0, 1.5, 5.0, 4.5, 2.5]
'J=', {(1, 2): 3.0, (2, 3): 3.0, (3, 4): 3.0, (0, 2): 3.0}
```

```
'ising_offset=', 3.5

embedding= [[0], [1], [4], [2], [7]]

Physical qubits used= 5
Attempting to connect to network...

new_emb= [[0], [1], [4], [2], [7]]
h1= [0.4, 0.3, 0.9, 0.0, 1.0, 0.0, 0.0, 0.5, 0.0, ...
j1= {(2, 7): 0.6, (2, 4): 0.6, (1, 4): 0.6, (0, 4): 0.6}

annealT= 20 progT= 100 readT= 100 num= 1000

result: {'timing': {'total_real_time': 490474,
'anneal_time_per_run': 20,
'post_processing_overhead_time': 548,
'readout_time_per_run': 309,
'total_post_processing_time': 548,
'run_time_chip': 470180},
'energies': [-2.5, -2.3, -1.9000000000000001],
'num_occurrences': [666, 333, 1]
'solutions': ...
}
newresult: [[-1, -1, 1, -1, 1], [1, 1, -1, 1, -1], [1, 1, -1, -1, 1]]

solution 0 size= 2 weight= 9.0 [0, 0, 1, 0, 1]
solution 1 size= 3 weight= 8.0 [1, 1, 0, 1, 0]
solution 2 size= 3 weight= 6.0 [1, 1, 0, 0, 1]
```