Proceedings of the Ninth International
Symposium on Combinatorial Search (SoCS 2016)

A Hybrid Quantum-Classical Approach to Solving Scheduling Problems

Tony T. Tran,[+,‡,†,*] **Minh Do,**[‡,†] **Eleanor G. Rieffel,**[+] **Jeremy Frank,**[‡]
Zhihui Wang,[+,**] **Bryan O'Gorman,**[+,†] **Davide Venturelli,**[+,**] and **J. Christopher Beck**[*]
[+] Quantum Artificial Intelligence Lab., NASA Ames Research Center, Moffett Field, CA
‡Intelligent Systems Division, NASA Ames Research Center, Moffett Field, CA
** Universities Space Research Association, Mountain View, CA
†Stinger Ghaffarian Technologies, Inc., Greenbelt, MD
*Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, ON

Abstract

An effective approach to solving complex problems is to decompose them and integrate dedicated solvers for those subproblems. We introduce a hybrid decomposition that incorporates: (1) a quantum annealer that samples from the configuration space of a relaxed problem to obtain strong candidate solutions, and (2) a classical processor that maintains a global search tree and enforces constraints on the relaxed components of the problem. Our framework is the first to use quantum annealing as part of a complete search. We consider variants of our approach with differing amounts of guidance from the quantum annealer. We empirically test our algorithm and compare the variants on problems from three scheduling domains: graph-coloring-type scheduling, simplified Mars Lander task scheduling, and airport runway scheduling. While we were only able to test on problems of small size, due to the limitation of currently available quantum annealing hardware, our empirical results show that information obtained from the quantum annealer can be used for more effective search node pruning and to improve node selection heuristics when compared to a standard classical approach.

Introduction

Quantum annealing is a heuristic quantum algorithm that solves combinatorial optimization problems on a quantum computational device. Quantum computing is a nascent technology without the decades of research and development that classical computers have experienced. Current quantum computational hardware of more than several qubits consists of only limited quantum annealers, special purpose quantum hardware designed to run the quantum annealing metaheuristic. Our goal is to extend the use of quantum annealing on currently available specialized quantum hardware through integration with classical computing using decomposition. We further explore a novel hybrid quantum-classical framework, based on tree search, that we introduced in (Tran et al. 2016). The quantum annealer samples from the configuration space of a relaxed problem to obtain strong candidate solutions, while the classical algorithms maintain a global search tree, as well as handling the relaxed components of the problem to check the validity and quality of the candidate solutions.

Our framework takes advantage of the strength of quantum annealing dedicated quantum hardware, and complements it with classical processing to enable the entire algorithm to be complete. Our framework is the first that uses quantum annealing as part of a complete search. Results returned by the quantum annealer guide exploration and pruning of the search tree. Specifically, we use the solution quality of samples returned by the quantum annealer from different parts of the configuration space to help guide further search. A novel aspect of this work is that our approach makes use of all results returned by the quantum annealer, not just the optimal ones.

We evaluate the performance of six node-selection heuristics, with progressively greater guidance from the results obtained from the quantum annealer, on instances from three scheduling domains: graph-coloring-type scheduling and airport runway scheduling, as well as Mars Lander task scheduling which we previously explored in (Tran et al. 2016). These three domains have varying complexity in their decompositions: (1) a decision problem that is fully represented on the quantum annealer, but gains completeness from our framework, (2) a decision problem in which the quantum annealer samples from the configuration space of a relaxation of the original problem, and (3) an optimization problem in which the classical algorithm reasons about the objective function.

This paper provide a proof of concept quantum-classical hybrid framework that uses quantum annealing to guide tree-search, and ensuring a systematic and complete search, not a competitive state-of-the-art approach to solving combinatorial problems. The main contributions of this work are:

- A novel framework for quantum-classical hybrid approaches to combinatorial problems.
- The use of quantum annealing in a complete search.
- An algorithm that makes use of all results returned by the quantum annealer, not just the best ones.

Quantum Annealing

Quantum computing enables more efficient solution to certain classes of problems than classical computing (Rieffel and Polak 2011; Nielsen and Chuang 2001). While large-scale universal quantum computers are likely decades away, special-purpose quantum computational devices are

emerging, making it possible to empirically evaluate heuristic quantum algorithms such as quantum annealing. Quantum annealers run quantum annealing (Farhi et al. 2000; Smelyanskiy et al. 2012), a metaheuristic algorithm that makes use of quantum tunneling and interference (Das and Chakrabarti 2008; Boixo et al. 2014). It is one of the most accessible quantum algorithms to people versed in classical computing because of its close ties to classical optimization algorithms such as simulated annealing and because the most basic aspects of the algorithm can be captured by a classical cost function and parameter setting.

A quantum annealer minimizes Quadratic Unconstrained Binary Optimization (QUBO) problems of the form

$$C(\mathbf{x}) = \sum_i c_i x_i + \sum_{i<j} c_{i,j} x_i x_j,$$

where $\{c_i, c_{i,j}\}$ are real coefficients and $\mathbf{x} \in \{0,1\}^n$. An application problem must be mapped to a QUBO problem to be solved on a quantum annealer. We describes QUBO mappings for the three targeted scheduling domains later.

Variables in the QUBO formulation are mapped to qubits on the hardware. Because the physical hardware has limited connectivity, it is often necessary to represent a single variable using multiple qubits (connected to each other in a subtree). These subtrees are chosen to ensure that each pair of variables appearing in a quadratic term in the QUBO are connected through a pair of qubits within their respective subtrees. Minor embedding is the process of determining which physical qubits will represent which variables (Choi 2011). O'Gorman et al. (2015) provide more details about embedding and the process of using a quantum annealer.

Quantum Annealing Guided Tree Search

We are interested in using our framework to enable the use of current quantum annealing hardware for scheduling problems where: (1) a complete search is desired, (2) the problem has properties that require more resources than available on the hardware, whether with respect to number of qubits, precision of coefficients, or both. To realize these objectives, we decompose problems so that the relaxed problem can be embedded on the quantum annealing hardware given its current limitations in terms of size, connectivity, and precision. The precise relaxation is problem dependent. In this section, we will present details of the general framework and provide domain specific details in later sections.

Our classical-quantum tree search algorithm (Figure 1) has three components: a global component that maintains the search tree and handles the problem decomposition, and two solvers, a quantum annealer to solve a relaxation of the problem and a solver run on a classical computer that considers the remaining portions of the problem. The global tree search manager decides when each sub-solver is called and what problem will be solved on that sub-solver.

The global search tree manager maintains a partial binary tree that is constructed from configurations found in the master problem. The manager then navigates through the tree systematically, identifying sub-spaces for which the quantum annealer and classical subproblem solver are called to further expand the tree.

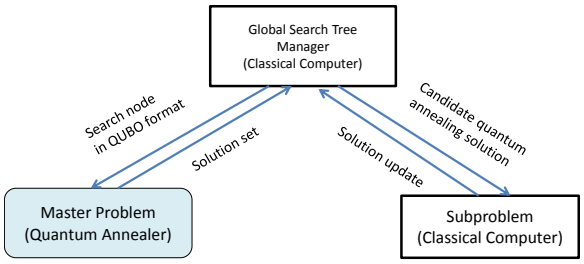

Figure 1: Tree-search based Quantum-Classical Algorithm.

Problem definition: Consider problems of the form:

$$\min \quad f(\mathbf{x}) \tag{1}$$
$$\text{s.t.} \quad \mathbf{x} \in \Phi, \tag{2}$$

Here, $\mathbf{x}$ is a vector of binary decision variables, f is a real-valued objective function, and Φ is the feasible space of $\mathbf{x}$ defined by the problem constraints. It is possible to represent this problem as an unconstrained optimization problem by moving the constraints into the objective function and penalizing assignments that break constraints. Let the penalties on the constraints be represented by a function $g(\mathbf{x})$, where $g(\mathbf{x}) = 0$ if $\mathbf{x} \in \Phi$ and $g(\mathbf{x}) > 0$ otherwise. Thus, finding

$$\min \quad f(\mathbf{x}) + \lambda g(\mathbf{x}) \tag{3}$$

is equivalent to solving the general problem described by (1) and (2), where λ is a constant used to ensure that constraint violations out-weigh the optimization function. If it is a decision problem, it can be represented as $\min g(\mathbf{x})$ where a configuration with $g(\mathbf{x}) = 0$ is a feasible solution. We consider (3) the global problem to be solved.

Problem decomposition: The quantum annealer will be used only on decision problems, i.e., $C(\mathbf{x}) = g(\mathbf{x})$. If the problem of interest is an optimization problem, we relax it by focusing on $g(\mathbf{x})$ and ignore $f(\mathbf{x})$. We can further relax the problem by removing constraints so that constraint (2) becomes $\mathbf{x} \in \bar{\Phi}$, where $\Phi \subset \bar{\Phi}$. The master problem is then

$$\min \quad \bar{g}(\mathbf{x}) \tag{4}$$

where $\bar{g}(\mathbf{x})$ are the penalties for violating constraints in the relaxed problem. It is not generally true that quantum annealers are unable to solve problem (3) directly; in practice, due to current hardware limitations on precision, i.e., limitations on the precision of the coupling strength between qubits that affect the range of the values that the coefficients can be set to in the objective function, it may not always be possible to define a penalty function $g(\mathbf{x})$ and multiplier λ that can be used on the quantum hardware, which still guarantees the set of feasible solutions will result in a lower cost solution than infeasible solutions. Furthermore, even if precision is not a problem, it may still not be possible to fit the decomposed problem on the hardware. In order to formulate constraints in the QUBO model, additional variables may be required. Even without the additional variables used to represent constraints, it is often the case that adding quadratic terms to the QUBO formulation greatly increases the connectivity of the resulting problem, which will increase the

Figure 2: A partial binary search tree example with open nodes indicated by shaded nodes and pruned nodes crossed out in red. The master problem objective function, $C(\mathbf{x})$, is presented in the node itself and subproblem solution presented as $[\cdot]$ below the nodes with $C(\mathbf{x}) = 0$. (Left) Tree after the first iteration through solving the master problem and subproblem. (Right) Tree after the second iteration through solving the master problem and subproblem, where node (B) was expanded.

number of required qubits in the embedded problem. Thus, to solve even small and medium sized problems, it is essential that problem (3) is relaxed to problem (4) in order to be solved on the quantum annealing hardware.

The subproblem is then to enforce the relaxed components of the global problem by calculating,

$$f(\mathbf{x}^*) + \lambda g(\mathbf{x}^*) \tag{5}$$

for a configuration $\mathbf{x}^*$ returned by the quantum annealer.

As an example, assume we wish to solve the problem,

$$\begin{aligned} \min \quad & 2x_1 + x_2 - 2x_3 && (6)\\ \text{s.t.} \quad & x_1 + x_2 + x_3 = 2, && (7)\\ & x_1 \geq x_2, && (8)\\ & x_i \in \{0,1\} \qquad i = 1,2,3. && (9) \end{aligned}$$

We can represent constraints (7) and (8) as,

$$g_1(\mathbf{x}) = (2 - x_1 - x_2 - x_3)^2,$$

and

$$g_2(\mathbf{x}) = x_2 - x_1 x_2,$$

respectively. If the constraints are satisfied, then $g_1(\mathbf{x}) + g_2(\mathbf{x}) = 0$. Otherwise, $g_1(\mathbf{x}) + g_2(\mathbf{x}) > 0$ and the solution is invalid. The original problem is an optimization problem, so the master problem is

$$\min \quad g_1(\mathbf{x}) + g_2(\mathbf{x}) \tag{10}$$

The subproblem is then to evaluate expression (5). In our example, we need only to evaluate $f(\mathbf{x}^*)$ in expression (5) since any feasible solution will have $g(\mathbf{x}^*) = 0$. However, in general this is not the case and it may be necessary to also evaluate $g(\mathbf{x})$ if some of the constraints were relaxed or removed for the master problem.

Solving the master problem: The quantum annealer runs the relaxed problem defined by (4) to obtain configurations to populate the global search tree. For each job submitted to the quantum annealer, K anneals are performed and $\bar{K} \leq K$ unique configurations of varying quality are returned. We denote the unique set of returned configurations as Ψ.

Continuing with our example, let us assume that the quantum annealer is called $K = 3$ times on problem (10) and three unique configurations are found for (x_1, x_2, x_3): $(0,0,0)$, $(0,0,1)$, and $(1,1,0)$. The calculated value of $C(\mathbf{x})$ for these three solutions are 4, 1, and 0, respectively. From these configurations, only $(1,1,0)$ is found to be valid with respect to constraints (7) and (8).

Solving the subproblem: Since the configurations found by the quantum annealer are solved only for the (relaxed) master problem, we must check that the relaxed constraints in the global problem are properly enforced when a configuration $\mathbf{x} \in \Psi$ is found with cost $C(\mathbf{x}) = 0$. Any configuration with greater cost is invalid for the global problem and so the subproblem does not need to be solved for these configurations. Thus, from our example, we see that only configuration $(1,1,0)$ requires the use of the subproblem solver. Using the returned quantum annealing configurations as input, the subproblem will calculate the expression (5) and send the result to the global tree manager to update to the correct values. For our example, the configuration $(1,1,0)$ evaluates to an objective value of $f(\mathbf{x}) = 3$. As this configuration satisfies all constraints ($C(\mathbf{x}) = 0$), we know that it is an incumbent solution with a cost of 3.

Building the partial tree and generating open nodes: Using the set of unique configurations Ψ returned by the quantum annealer, build a partial binary tree with a fixed variable ordering. The left (right) branch of a node corresponds to the variable being set to 0 (1) in Ψ (see Figure 2). The cost $C(\mathbf{x})$ of a configuration is shown inside the corresponding node. Where a configuration $\mathbf{x}^*$ is feasible, i.e., $C(\mathbf{x}^*) = 0$, the true objective function $f(\mathbf{x}^*)$ is presented in brackets below the node. The partial tree is traversed to generate all open nodes (the shaded nodes), nodes that have not yet been explored but have a parent node that is already expanded.

Node pruning: Open nodes are pruned by inference algorithms based on the problem-specific constraints and objective function. At any open node (shaded nodes), a subset of decision variables have been set. Based on this partial configuration, a check is performed to see whether any constraints are violated. For example, consider a bounding function $h(\bar{\mathbf{x}}) = \min_{\hat{\mathbf{x}}} g(\bar{\mathbf{x}})$, where $\bar{\mathbf{x}}$ is the set of instantiated variables and $\hat{\mathbf{x}}$ is the set of variables still to be decided at open node N. If $h(\bar{\mathbf{x}}) > 0$, then the node N is pruned. This process is a standard inference technique in constraint programming that is sound, but incomplete. The same can be done for $f(x)$ to remove partial configurations that would

lead to sub-optimal solutions when the bounding function on the objective is greater than the current incumbent solution objective. This approach is similar to the bounding that is found in standard branch-and-bound approaches.

If no constraints have been violated, then a forward checking procedure (Haralick and Elliott 1980) is performed. It is often the case that some uninstantiated variables have a single feasible value remaining based on the partial configuration, further simplifying the problem at the open node. However, our current approach maintains a static ordering of variables in the tree. Thus, forward checking is only useful if inferences can be made on the next variables. Using a dynamic ordering could enable more effective pruning using forward checking, but is not considered in this initial study.

Based on our running example, we must evaluate each open nodes to check whether it can be pruned. We check if a constraint has been violated or if a lower bound on the objective function is worse than the incumbent solution objective currently equal to 3. Open node (A) corresponds to the partial configuration (0, 1) and this node is pruned as the partial solution violates constraint (8). Node (B), corresponding to partial configuration (1, 0), does not violate any constraints. We can also calculate a simple lower bound by instantiating all commited variables in the partial configuration, and then including the value of any negatives terms in the objective function (6) associated to the uncommitted variables. Thus for node (B), we can first calculate the cost of the partial solution so far to be $2 \cdot 1 + 1 \cdot 0 = 2$, and then subtract 2 as it is still possible to set $x_3 = 1$ and reduce the objective function. Therefore, open node (B) has a lower bound of 0, which is not pruned as the current incumbent is 3. Finally, open node (C) corresponds to the configuration (1,1,1), which is pruned since the solution violates constraint (7).

Node selection and exploration: Once the partial tree is built and open nodes are generated, unless the tree has been fully explored, search must continue. Since open nodes represent sub-spaces that have not been a part of any configuration found for the master problem, the global tree search manager will select one of these nodes to explore next. Exploration of an open node means that the master problem is invoked at the particular location in the tree. Since we wish to start the search from a partial configuration, the QUBO is updated with $\bar{\mathbf{x}}$ and the rest of the decision variables $\hat{\mathbf{x}}$ are solved for in the master problem. At the node, the master problem and subproblem are solved again to create a new partial sub-tree and the process is repeated.

In our running example, the only open node still to be considered is node (B). However, in general, one can expect many more open nodes to be present. These nodes are to be given some heuristic ordering for expansion. To expand node (B), which has a partial configuration already, we must update the master problem. The resulting master problem after the correct values for x_1 and x_2 are set is,

$$\min \quad g(\mathbf{x}) = 1 - x_3. \tag{11}$$

The master problem is now significantly simpler as most decisions are already made and the only decision left is x_3. New configurations are found for this master problem and are used to build the tree below node (B).

Algorithm 1 Quantum Annealing Guided Tree Search.

```
open_nodes: a priority queue
push root_node to open_nodes
while open_nodes ≠ NULL do
    pop n from open_nodes
    Ψ =solve_master_problem(n)
    for x ∈ Ψ do
        if quantum_feasible(x) then
            solve_subproblem(x)
    build_partial_tree(Ψ)
    new_open_nodes = generate_open_node(Ψ)
    push new_open_nodes to open_nodes
    prune(open_nodes)
```

In our study, the ordering heuristic we consider uses the weighted sum of two node selection heuristics based on: *slack* and *quantum annealer's configuration quality*. The slack measure S at an open node quantifies the extent of available options for the remaining decisions to be made. We define $S = \left(\prod_{j \in J'} d_j\right)^{\frac{1}{|J'|}}$, where J' is a set of decisions to be made and d_j is a domain-dependent measure of the remaining domain size of a decision $j \in J'$. The domain size and definition of J' will be expanded upon in later sections. The *quantum annealer's configuration quality* measure is calculated using the quality of configurations returned by the quantum annealer at an open node as: $C^* = \min_{\mathbf{x} \in Y} C(\mathbf{x})$, where the set Y contains all the configurations found so far below the sibling node of the open node. We examine the performance under different weightings of the two heuristics, $V = (1 - \alpha)S - \alpha C^*$, for various values of α. Open nodes with the highest value of V will be explored first.

Conditions for Termination: For decision problems, once a feasible solution has been found or the open node list is empty, the search is terminated. For optimization problems, the search is terminated as soon as the open node list is empty, indicating that all nodes have been either explored or pruned.

Algorithm 1 presents the pseudo-code for our framework. Starting from the root node, the master problem solver is called on the relaxed problem to obtain a set of configurations Ψ. If a configuration $\mathbf{x} \in \Psi$ is feasible for the master problem, i.e., $C(\mathbf{x}) = 0$, then the subproblem is solved given the master problem solution $\mathbf{x}$. The partial tree is then built using Ψ, at which point open nodes are generated and added to the open node list. All the nodes on the open node list are checked for consistency and pruned if found to be inconsistent. The process is then repeated for a node on the open node list until no open nodes remain.

Scheduling Domain Description

We tested our approach on three scheduling domains.

Graph-Coloring-Type Scheduling

Consider a scheduling problem with a set of tasks and constraints that any pair of tasks competing for the same re-

source cannot be assigned the same time-slot. Such scheduling problems can be viewed as *vertex coloring* problems by representing tasks as vertices, resource contention between tasks as edges, and time slots as colors. Given a graph $G = (V, E)$, with vertices V and edges E, a κ-coloring problem assigns one of κ colors to each vertex in such a way that no two vertices that share an edge have the same color. We consider the decision problem with $\kappa = 3$: given a graph G, is there a 3-coloring of G?

Mars Lander Task Scheduling

The simplified Mars lander domain consists of tasks that the lander must perform in the course of a Martian day. Each task requires full use of the Mars lander for some duration and consumes battery power at a task-dependent rate. Solar panels on the Mars lander recharge the battery concurrently while the lander performs other tasks. Each problem consists of a set of tasks $j \in J$, each with processing time and set of time points H_j at which it may be scheduled, contained within a Martian day of length H. Tasks consume power from an onboard battery at a rate of e_j per time point while being processed. The battery has maximum, $E_{\max}$, and minimum, $E_{\min}$, levels. Solar panels recharge the amount e_t^+ during time point t, which depends on available sunlight. The goal is to construct a feasible schedule, one that assigns each task a start time, adhering to the tasks' time-windows, precedence, and battery constraints.

Airport Runway Scheduling

The airport runway scheduling problem Gupta, Malik, and Jung (2009) consists of a set of aircraft, F, approaching a single runway, where each aircraft $j \in F$ enters a queue, $q \in Q$ to be scheduled for take-off. Each aircraft is already pre-assigned to a specific queue, q_j, and cannot take-off until all aircraft ahead of it in the same queue have left. Each aircraft belongs to one of four size categories, which determines the minimum separation required between consecutive departures. Between any two aircraft i and j, there must be a minimum separation time of d_{ij}, dependent on the size categories of those two aircraft. A feasible schedule of the problem is an assignment of a take-off time z_j to each aircraft satisfying all constraints. We target the optimization problem of minimizing the objective function $\sum_{j \in F} z_j$.

Mapping scheduling problems to QUBO

The relaxed scheduling problems for the three different domains have common features that enable us to discuss the mapping of these problems to QUBO problems in a unified way. We leave details specific to each relaxed problem to the next section. The scheduling problems we consider involve *tasks* to schedule and discrete *resources* (e.g., time slots). In the QUBO formulations, a binary variable $x_{j,r}$ indicates whether task j is assigned resource r.

Constraints are captured by penalty terms in the QUBO cost function. Two basic types of constraints must be captured in all scheduling problems:

1. **Unique assignment**: this penalty term enforces that each task j must be assigned exactly one resource:

$$C_j^{\text{unique}} = (\sum_r x_{j,r} - 1)^2. \tag{12}$$

2. **Exclusive use of resource**: this penalty term enforces that assigning resource r to task j precludes assigning certain resources to another task j'

$$C_{j,r,j'}^{\text{exclusive}} = \sum_{r' \in R_{j,r,j'}} x_{j,r} \times x_{j',r'} \tag{13}$$

where $R_{j,r,j'}$ is the set of resources that become inaccessible to task j' if resource r is assigned to task j.

Which resources become inaccessible varies in different domains, resulting in slightly different $C_{j,j'}^{\text{exclusive}}$ terms:

- Color difference (Graph Coloring): if two vertices j and j' are connected by an edge, they cannot share the same color. Thus, $x_{j,r} \times x_{j',r}$ is added for all colors r.
- Precedence relations (Mars-Lander and Airport Runway): if task j is supposed to precede task j', penalty term $x_{j,r} \times x_{j',r'}$ is added for any time $r > r'$.
- Job length (Mars-Lander): during each job j execution, no other job j' can start. Thus, $x_{j,r} \times x_{j',r+\tau}$ is added for $\tau = [0, p_j)$, where p_j is the processing time of j.
- Flight separation (Airport Runway): successive flights require a minimum separation time between their departures, which is dependent on the sizes of the two aircrafts. Thus, $x_{j,r} \times x_{j',r+\delta}$ is added for $\delta \in [0, d_{j,j'})$.

The final QUBO cost function is thus

$$C = \sum_{j \in J} C_j^{\text{unique}} + \sum_{j,j' \in J} \sum_r C_{j,r,j'}^{\text{exclusive}}, \tag{14}$$

where J is the set of all tasks. A feasible solution has $C = 0$. If $C > 0$, at least one constraint is violated.

Domain-Specific Details

The three domains suggest quite different decompositions: (1) no decomposition, (2) decomposition of a decision problem into a simpler decision problem and a constraint-check and (3) decomposition of an optimization problem into a decision problem and an optimization problem.

Graph Coloring

Problem Decomposition: The full graph coloring problem is mapped to a QUBO and run on the quantum annealer with no decomposition. The classical processor is used only to prune and maintain the tree search and to compute which node to try next using the node selection heuristic.

Node Pruning, Propogation and Selection Metric: We prune a node from the tree when a classical check determines that the current partial configuration removes all possible colors for an uncolored vertex. This *forward checking* procedure ensures that each remaining decision variables $\tilde{\mathbf{x}}$ has at least one value that is consistent with the assignment

to $\bar{\mathbf{x}}$. For every vertex not yet colored, we check all neighboring colored vertices and remove those colors from the uncolored vertex's domain. Any vertex left with an empty potential color set open node is pruned. If the next vertex to be colored in the sequence defined by our tree has only one color available in its domain or has already been colored, the remaining variables associated to the vertex are assigned appropriately and the open node is updated to a deeper node corresponding to the new partial solution. The alternative branches between the original open node and the new open node do not require exploration because these branches lead to infeasible colorings. The size d_j of the color domain for vertex j is used to compute the slack.

Problem generation: Following (Rieffel et al. 2015), we extended Culberson et al.'s graph generator program (Culberson, Beacham, and Papp 1995) to generate 20 Erdös-Rényi graphs at the colorable-uncolorable phase transition with 16 nodes that have a feasible 3-coloring.

Mars Lander Task Scheduling

Problem Decomposition: The subproblem handed to the quantum annealer ignores battery constraints. The classical processor must check whether the battery level is violated at any time in any configuration returned by the quantum annealer with $C(\mathbf{x}) = 0$,

Node Pruning, Propogation and Selection Metric: For each unscheduled task j in a partial solution, we remove from H_j all task starting times which would conflict with an already scheduled task. We define the cardinality of the remaining potential starting times for j as d_j and prune any open node with an unscheduled task that has $d_j = 0$. If the next unscheduled task has $d_j = 1$, the partial solution at the open node is appended to schedule the job at the remaining time point and the deeper node is used instead. Similarly, if a task j has already been scheduled, i.e., $x_{j,r} = 1$ for some time point r, but the partial solution does not assign values for all variables associated to task j, the open node can be updated accordingly by setting all remaining $x_{j,r}$ values to 0 and continuing exploration at the deeper node. Furthermore, if any of the resulting scheduled tasks conflict with each other, the node is pruned.

Problem generation: We consider problems with six tasks, each of which is based on actual tasks performed by the Phoenix Mars lander, but the detailed values of the durations (between 0.5 and 2 hours), time-windows (task-dependent, within a 16-hour horizon), and battery consumption rate (between 3-11% per 0.5 hours) are fabricated. Only the initial and maximum battery charges and charging rates were varied across the 21 instances we considered.

Airport Runway Scheduling

Problem Decomposition: The quantum annealer decides whether or not there is a valid runway schedule and the classical processor calculates the objective function of minimizing the total take-off time for all aircraft.

Node Pruning, Propogation and Selection Metric: As in the Mars Lander case, for each node j, we remove variables for slots that conflict with the departure times of scheduled aircraft, and calculate the cardinality d_j of the set of remaining available take-off times for flight j. Open nodes are updated using the same propogation mechanisms as for the Mars Lander problem. The classical processor then calculates the objective function, and keeps track of the best solution found so far. The classical processor also computes, for each open node, a lower bound by summing the start times of all the already scheduled departures in the partial schedule. For each unscheduled departure, the earliest start is determined, and is added to the lower bound. If this lower bound is greater than the cost of the best solution found so far, the open node is pruned.

Problem Generation: We generate problem instances similar to Gupta, Malik, and Jung (2009), but with reduced size in order to obtain instances that can be fit on to the D-Wave 2X hardware. Specifically, we generate 20 problem instances with 6 aircraft that arrive during a 5 minute time-period, with time discretized into 30 second segments. Arrival times are randomly generated following a uniform distribution. Each aircraft will enter one of three queues. All flights are to depart within a 7.5 minute horizon. We ensure all instances can be embedded on the hardware and have feasible solutions since we are interested in how well the hybrid decomposition is able to find and prove optimality.

Empirical Evaluation and Analysis

We present experimental results on the three scheduling domains described in the previous sections. In each case, the computational effort is given in terms of how many times an open node is expanded for exploration and the number of unique configurations found when the algorithm terminates. We provide tables that present the average values of these metrics, as well as figures showing the median, 35th, and 65th percentiles. These numbers are obtained for each of the six values of the node search weight α that we tried. In addition, we include the average performance metric of a variant denoted as Random-Sample. This variant randomly samples configurations rather than makes use of the quantum annealer and was studied to verify that the quantum annealer is in fact optimizing for the QUBO problem properly.

Running on the D-Wave 2X Quantum Annealer

We implemented our code in Python, using D-Wave's Python API to interface with the D-Wave 2X machine. Each time the quantum annealer is invoked at an open node, $K = 10,000$ anneals are performed, each with an anneal time of 20 micro-seconds. Embedding and parameter setting for the embedded QUBO are done using D-Wave's software with default parameters (Cai, Macready, and Roy 2014) except for setting the coupling strength between physical qubits representing the same variable. Based on results in (Rieffel et al. 2015), we set this coupling strength to 1.4 for the graph coloring problem. For the other problems, our own experimentation suggested a coupling strength of 5.0.

When running on the D-Wave 2x machine, jobs are submitted to a queue while awaiting processing. As such, we do not have immediate access to the hardware and must therefore idle between iterations of the master problem and subproblem. Although some aspects of runtime are available,

it is currently not possible to obtain a truly accurate runtime for the quantum annealing process as part of our framework, especially so if we were to consider having dedicated hardware for our own use. We know the anneal time, which can act as a lower bound on time, and the wall clock time from when we initiate a call for the quantum annealer until the time when the configurations are returned, but neither of these are an accurate measurement for the effort required to use the quantum annealer as part of a hybrid framework. The current bottleneck of our implementation is waiting for the results from the quantum annealer. The time spent performing other processes such as solving the subproblem, building the tree, and pruning nodes are negligible. We do not expect that these processes are negligible in general, but for the problems studied in this paper, these aspects account for a minuscule fraction of the actual runtime.

Graph Coloring

Each problem instance has 48 logical variables, which results in embedding sizes of 125 to 310 qubits. The computational results for the different algorithm variants are presented in Figure 3a and Table 1. The results show that using the configuration quality of the solutions provided by the quantum annealer can improve the performance over the slack-only heuristic when sufficiently large α values are chosen. The results on the number of open nodes explored suggest that balancing α may be important in this domain, since $\alpha = 0.4$ providing the best results. In particular, α value of 0.2 leads to worse performance than Slack-Only, but with larger α values the performance improves.

Mars Lander

The QUBO for the master problem common to all 21 Mars lander task scheduling problems contains 52 variables, and is embedded in 764 qubits. The computational effort results presented in Figure 3b and Table 1 show that, as we found in the graph coloring domain, search node selections guided by solutions returned by the quantum annealer improves performance over the slack-only heuristic. Unlike in the graph coloring domain, we see that any $\alpha > 0$ chosen improves performance over the slack-only condition. Of the different α values tested, the range $0.4 \leq \alpha \leq 1.0$ yields the best performance. Further investigation would be needed to distinguish differences in performance in this range.

Airport Runway

The QUBOs for the airport runway scheduling problem instances all contain between 31 and 50 logical variables, with embedded sizes between 645 and 981 qubits. The results are presented in Figure 3c and Table 1. The median performance shows no particularly strong trend when varying α. However, the mean performance as presented in Table 1 suggests that guiding the search does improve average performance. The difference between the results for the mean and median indicate that the Slack-Only case, and to a lesser extent the QA-Only variant, have heavy tails, with a few instances performing very poorly.

The less definitive results in this domain stem from a significant difference between this domain and the previous two in which the problem is to find a feasible solution, whereas the airport runway scheduling problem asks for an optimal solution. The node selection heuristic does not take the objective function into account and so the heuristic is ignorant of any information about the objective function when deciding where to search next in the tree. For this reason, differences in these node selection heuristics tested may be harder to detect because node selection itself has less of an effect on the computational effort.

Summary: Our results show that this hybrid framework can handle a variety of scheduling domains with different objective functions and decomposition strategies. For the decision problems, the results obtained from the quantum annealer not only focus the search by enabling effective pruning, but also contribute positively to improved search node selection. The same improvements are not as apparent for the optimization problem, but using more sophisticated search techniques within our framework would improve performance.

A tree search framework has long been recognized as an effective methodology for solving combinatorial problems. Table 1 shows that even using a random sampler within the tree search framework is sufficient to solve these combinatorial problems, as the pruning and node selection heuristics are sufficient in guiding search even though the master problem solver used is ignorant to the problem. Introducing the quantum annealer improves search by providing better guidance for the traversal through tree.

Related Work

Given the relative novelty of quantum annealing hardware, research in this area has been limited. (Rieffel et al. 2015; Venturelli, Marchand, and Rojo 2015). have expored pure quantum annealing approaches for some planning and scheduling problems. Instead of using the quantum annealer to optimize, Benedetti et al. (2015), and Adachi and Henderson (2015) explore the possibility of using it as a Boltzmann sampler to aid the training phase in deep learning, quite a different use of sampling compared to our approach.

Combining quantum and classical computing in algorithms have only recently begun being explored. Rosenberg et al. (2015) present a large-neighbourhood local search with a method to integrate the quantum annealer as a subroutine within a classical algorithm. In a similar fashion, Zintchenko, Hastings, and Troyer (2015) propose a hierarchical search that decomposes the set of decisions variables into multiple groups and cycles through groups optimizing the sub-problem of a particular group while fixing all other variables, performing quantum annealing on each group. However, both of these studies do not implement the algorithm on a quantum annealer. Rosenberg et al. present results using a tabu-search and Zintchenko, Hastings, and Troyer use simulated annealing in place of quantum annealing. Importantly, our work is distinguished from these in that our approach performs a complete search.

Conclusion and Future Work

We presented a tree-search based quantum-classical framework in which results from a quantum annealer prune and

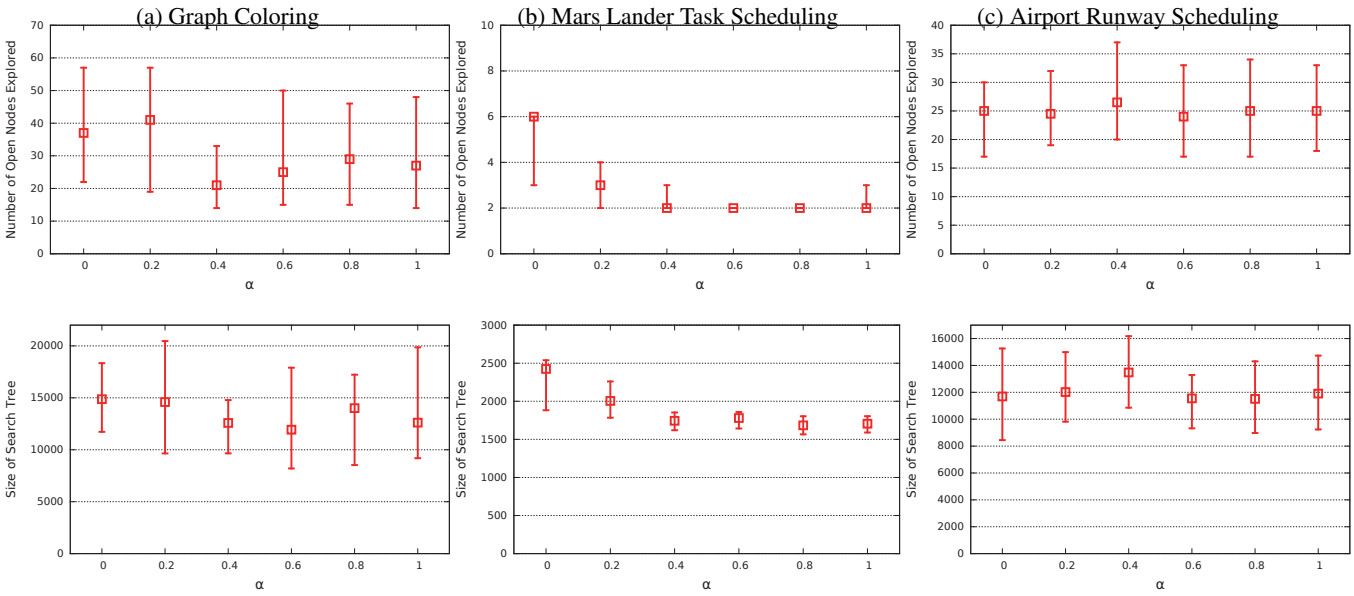

Figure 3: Results for the algorithm variants on all problem instances considered: solving each instance ten times for each variant. The median size of the number of open nodes explored (top) and the size of the search tree (bottom) is shown, with error bars at the 35th and 65th percentiles. Here, search tree size refers to the number of unique leaf nodes (configurations) found.

	Graph Coloring		**Mars Lander**		**Airport Runway**	
	avg. # of open nodes explored	avg. # of configurations found	avg. # of open nodes explored	avg. # of configurations found	avg. # of open nodes explored	avg. # of configurations found
Random-Sample	190.55	1,653,772.67	420.27	4,088,269.03	247.56	2,285,348.67
Slack-Only	58.99	22,269.95	4.10	2,143.21	45.72	15,858.21
Weighted (Best α)	38.78	16,858.30	2.22	1,728.33	31.58	13,488.71
QA-Only	68.06	23,790.14	3.25	1,822.60	35.54	14,643.77

Table 1: Mean performance for the algorithm variants on the problem instances considered: solving each instance ten times for each variant. We use the results from the best α value for the Weighted variant; these values are 0.4, 0.8, and 0.6, for the graph coloring, Mars lander, and airport runway scheduling problems, respectively.

guide the search. We tested multiple node-selection heuristics on instances from three different scheduling domains. The framework enables the use of a stochastic quantum-annealing solver within a complete search framework. Our results show that quantum annealer output can effectively prune and guide the search process in two of our tested domains, suggesting that further exploration of node selection metrics incorporating quantum annealing results is warranted, including other metrics of different form.

Our approach is not limited to strictly quantum-classical algorithms. The framework can be applied as a pure classical decomposition, allowing the use of specialized heuristic solvers for large and difficult problems.

In general, we do not expect quantum annealers to be competitive in the near-term against classical computing, with its decades-long headstart on research and development. Thus, one would expect that a classical-classical decomposition would outperform a quantum-classical decomposition. Our motivation in this work is to expand on the capabilities of the current quantum annealers, to provide a framework in which the benefits of even mature quantum annealing technology would be complemented by classical methods to obtain completeness and improve performance.

A potential extension of this work is to incorporate additional classical heuristics into the tree search, borrowing ideas from the extensive classical literature, such as variable ordering (Beck, Prosser, and Wallace 2004), conflict analysis and cutting planes (Achterberg 2007; Kelley 1960), and discrepancy-based search (Harvey and Ginsberg 1995; Beck and Perron 2000). One challenge to incorporating these ideas is keeping the resulting problems small enough that they can be run on current or near-term quantum annealers. Additional variables would need to be added to the QUBO formulation in order to incorporate constraints from cutting planes or no-good cuts. For this reason, it is interesting to explore the design of such extensions within the restrictions of quantum annealers.

Another extension is the application of improved pruning

and selection algorithms within the our framework to obtain better performance. Various inference methods for defining bounding functions can be designed for particular problem domains. Our framework lends itself to techniques found in constraint programming (Van Hentenryck, Simonis, and Dincbas 1992) and could benefit from the sophisticated inference algorithms developed in that literature. Using a dynamic ordering to improve the effect of forward checking is not currently performed, but is left as an area of improvement that can be implemented to enhance performance.

There remains much to learn about quantum annealing and the interplay between classical and quantum approaches. This work is an early step in an ongoing effort to provide insights into how to design and use special-purpose quantum hardware in service of practical applications.

Acknowledgements

The authors would like to acknowledge support from the NASA Advanced Exploration Systems program and NASA Ames Research Center. This work was supported in part by the AFRL Information Directorate under grant F4HBKC4162G001, the Office of the Director of National Intelligence (ODNI), and the Intelligence Advanced Research Projects Activity (IARPA), via IAA 145483. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, AFRL, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright annotation thereon.

References

Achterberg, T. 2007. Conflict analysis in mixed integer programming. *Discrete Optimization* 4(1):4–20.

Adachi, S. H., and Henderson, M. P. 2015. Application of quantum annealing to training of deep neural networks. *arXiv:1510.06356*.

Beck, J. C., and Perron, L. 2000. Discrepancy-bounded depth first search. In *Proceedings of the Second International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR 2000)*, 8–10.

Beck, J. C.; Prosser, P.; and Wallace, R. J. 2004. Trying again to fail-first. In *Recent Advances in Constraints*. Springer. 41–55.

Benedetti, M.; Realpe-Gómez, J.; Biswas, R.; and Perdomo-Ortiz, A. 2015. Estimation of effective temperatures in a quantum annealer and its impact in sampling applications: A case study towards deep learning applications. *arXiv:1510.07611*.

Boixo, S.; Smelyanskiy, V. N.; Shabani, A.; Isakov, S. V.; Dykman, M.; Denchev, V. S.; Amin, M.; Smirnov, A.; Mohseni, M.; and Neven, H. 2014. Computational role of collective tunneling in a quantum annealer. *arXiv:1411.4036*.

Cai, J.; Macready, W. G.; and Roy, A. 2014. A practical heuristic for finding graph minors. *arXiv:1406.2741*.

Choi, V. 2011. Minor-embedding in adiabatic quantum computation: II. minor-universal graph design. *Quantum Information Processing* 10(3):343–353.

Culberson, J.; Beacham, A.; and Papp, D. 1995. Hiding our colors. In *CP95 Workshop on Studying and Solving Really Hard Problems*, 31–42. Citeseer.

Das, A., and Chakrabarti, B. K. 2008. Colloquium: Quantum annealing and analog quantum computation. *Rev. Mod. Phys.* 80:1061–1081.

Farhi, E.; Goldstone, J.; Gutmann, S.; and Sipser, M. 2000. Quantum computation by adiabatic evolution. arXiv:quant-ph/0001106.

Gupta, G.; Malik, W.; and Jung, Y. C. 2009. A mixed integer linear program for airport departure scheduling. In *9th AIAA aviation technology, integration, and operations conference (ATIO)*, 21–23.

Haralick, R. M., and Elliott, G. L. 1980. Increasing tree search efficiency for constraint satisfaction problems. *Artificial intelligence* 14(3):263–313.

Harvey, W. D., and Ginsberg, M. L. 1995. Limited discrepancy search. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 607–615.

Kelley, J. 1960. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics* 703–712.

Nielsen, M., and Chuang, I. 2001. *Quantum Computing and Quantum Information*. Cambridge: Cambridge University Press.

O'Gorman, B.; Rieffel, E. G.; Do, M.; Venturelli, D.; and Frank, J. 2015. Compiling planning into quantum optimization problems: a comparative study. *Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS-15)* 11.

Rieffel, E. G., and Polak, W. 2011. *A Gentle Introduction to Quantum Computing*. Cambridge, MA: MIT Press.

Rieffel, E. G.; Venturelli, D.; O'Gorman, B.; Do, M. B.; Prystay, E. M.; and Smelyanskiy, V. N. 2015. A case study in programming a quantum annealer for hard operational planning problems. *Quantum Information Processing* 14(1):1–36.

Rosenberg, G.; Vazifeh, M.; Woods, B.; and Haber, E. 2015. Building an iterative heuristic solver for a quantum annealer. *arXiv preprint arXiv:1507.07605*.

Smelyanskiy, V. N.; Rieffel, E. G.; Knysh, S. I.; Williams, C. P.; Johnson, M. W.; Thom, M. C.; Macready, W. G.; and Pudenz, K. L. 2012. A near-term quantum computing approach for hard computational problems in space exploration. arXiv:1204.2821.

Tran, T. T.; Wang, Z.; Do, M.; Rieffel, E. G.; Frank, J.; O'Gorman, B.; Venturelli, D.; and Beck, J. C. 2016. Explorations of quantum-classical approaches to scheduling a mars lander activity problem. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*.

Van Hentenryck, P.; Simonis, H.; and Dincbas, M. 1992. Constraint satisfaction using constraint logic programming. *Artificial intelligence* 58(1):113–159.

Venturelli, D.; Marchand, D. J.; and Rojo, G. 2015. Quantum annealing implementation of job-shop scheduling. *arXiv preprint arXiv:1506.08479*.

Zintchenko, I.; Hastings, M. B.; and Troyer, M. 2015. From local to global ground states in Ising spin glasses. *Physical Review B* 91(2):024201.