Explorations of Quantum-Classical Approaches to Scheduling a Mars Lander Activity Problem

Tony T. Tran[+,‡,†,*], Zhihui Wang[+,], Minh Do[‡,†], Eleanor G. Rieffel[+],**
Jeremy Frank[‡], 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 problems involving mixed (continuous and discrete) variables and constraints, such as hybrid systems, is to decompose them into subproblems and integrate dedicated solvers geared toward those subproblems. Here, we introduce a new framework based on a tree search algorithm to solve hybrid discrete-continuous problems that incorporates: (1) a quantum annealer that samples from the configuration space for the discrete portion and provides information about the quality of the samples, and (2) a classical computer that makes use of information from the quantum annealer to prune and focus the search as well as check a continuous constraint. We consider four variants of our algorithm, each with progressively more guidance from the results provided by the quantum annealer. We empirically test our algorithm and compare the variants on a simplified Mars Lander task scheduling problem. Variants with more guidance from the quantum annealer have better performance.

Introduction

One approach to solving complex problems is to decompose them into subproblems that can be handled by dedicated solvers. Such an approach is particularly effective for large problems that involve different types, such as discrete and continuous, of variables and constraints. The key technical issue in such an approach is developing a framework to effectively decompose the problem and integrate the solvers.

Here, we introduce a new framework based on a tree search algorithm to solve hybrid discrete-continuous problems that incorporates:

- a quantum annealer that samples from the configuration space for the discrete portion and provides information about their quality
- a classical computer that (1) makes use of information from the quantum annealer to prune and hone the search; and (2) checks the continuous constraints on results returned by the quantum annealer.

Our framework takes advantage of the strength of quantum annealing, a metaheuristic for combinatorial optimization, and complements it with classical processing that enables the entire algorithm to be complete. As a result, this enables larger problems to be solved than is possible on current quantum annealing hardware. We evaluated our framework on a simplified Mars lander task scheduling problem that includes a continuous battery constraint and discrete requirements on the tasks to be scheduled. To keep the problem size manageable for the quantum annealer, we use it to search for schedules that meet the discrete requirements, and then use the classical computer to check the continuous requirements.

The results returned by the quantum annealer can be used for two different purposes: (1) prune the high-level search tree; and (2) guide which part of the tree to explore next using quality of results. Within our framework, we compare two different heuristics to guide the tree search:

1. node selection based only on the estimation by the classical computer without incorporating a quality measure for the results returned by the quantum annealer.
2. node selection incorporating information about the quality of the results returned by the quantum annealer.

The intuition behind the second heuristic is that parts of the tree that have returned better results for the discrete subproblem are more promising places to explore than parts that have returned worse results.

We compare specific settings of these two heuristics on instances of the Mars lander activity scheduling problem. For the quantum annealing component, we ran on the D-Wave 2X machine housed at the NASA Ames Research Center. Each run of a quantum annealer returns a pre-determined number of *configurations*, variable assignments to all of the variables in the cost function for the subproblem run on the annealer. We reserve the phrase *candidate solution* for feasible solutions to the subproblem and *global solution* for feasible solutions to the full problem. We also compare with a baseline approach in which instead of using the quantum annealer, random configurations are generated and used to prune the tree.

The main contributions of this work are:

- A novel framework for quantum-classical approaches to optimization problems that iteratively concentrates first on the discrete aspects of the problem and then on the continuous constraints.
- Instantiations of this framework that make use of a quantum annealer to sample the search space and guide future searches.
- Instantiations that make use of all configurations returned by the quantum annealer, not just the best configurations.

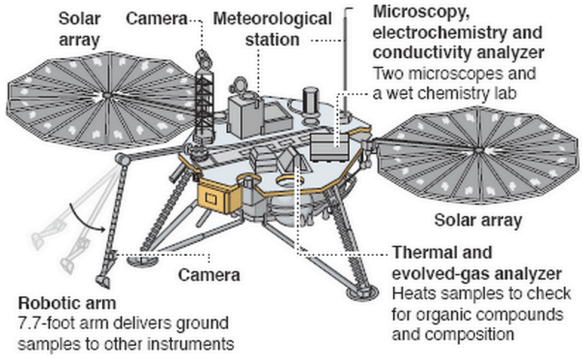

Figure 1: NASA's Phoenix Mars Lander. Source: NASA

- Full integration of a quantum annealer and a classical algorithm.
- Empirical results comparing different instantiations of this framework, with each other, and with a baseline.
- Established that feasible solutions can be found with less effort when the search is guided using configurations returned by the quantum annealer.

While our work is in an early stage and the scale of the problems we tested is limited, this decomposition framework also supports many other instantiations of these quantum-classical algorithms. We intend to explore other instantiations in future work.

The Mars Lander Problem

As a testbed to explore quantum-classical approaches, we consider a Mars lander that is tasked to perform multiple activities over the course of a Martian day. The Mars lander robotic spacecraft can land on the surface of Mars, but is not mobile like a rover. The lander has various scientific instruments and a robotic arm that can interact with its environment. Its activities include (1) scientific studies to achieve mission goals, (2) communication of data, and (3) operations to maintain the lander in a functioning state. The number of requested tasks can be large.

We consider a simplified application with a shorter scheduling horizon than a typical Martian day and fewer tasks. The scientific activities we consider are:

- **Obtain panoramic pictures:** Panoramic pictures of Mars landscape requires a *time-window* when there is sunlight, but also must take into account the strength and direction of the sun due to shadows and glare.
- **Measure Martian weather:** Measuring the Martian weather can have *time-windows* since scientists may be interested in measurements of particular times as the conditions change over the course of a day.
- **Sample Martian soil:** Sampling of Martian soil is subdivided into three different tasks: (1) taking a picture of the workspace, (2) digging, and (3) baking. The precedence constraints mean that digging can only occur once a picture of the workstation is taken and baking a sample only after soil is retrieved via digging.

The Mars lander will also need to **send stored data** via communication satellites when it has unobstructed line-of-sight to these satellites. Thus, there are only several disjoint time-windows in which an uplink task can occur.

In addition to the time-window and precedence constraints, the Mars lander has a limited-capacity battery and performing tasks depletes the battery at different rates. To ensure that there is enough power, the Mars lander is equipped with solar panels that **recharge the battery** when the sun is visible. The solar panel can be used at any time that the sun is visible, but the amount of power will depend on the amount of light which varies with weather conditions, time of day, and time of year. If the battery is at its maximum capacity, the excess power production from the sun cannot be stored. However, it is possible for the lander to draw power directly from the solar panels to power tasks rather than from the battery. This allows the lander to utilize power from the solar panels when the battery is fully charged rather than wasting it.

In our simplified problem, the Mars lander is capable of performing only a single task at a time. The only operation that can be done in parallel with other tasks is solar panel charging, which occurs automatically when the sunlight and battery capacity conditions are met. The goal is to construct a schedule that assigns each task a start time, adhering to the tasks' time-windows, precedence and battery constraints.

Problem Details: The parameters chosen here are artificially generated for the purposes of providing a problem inspired by the real NASA's Phoenix Mars lander problem. The scheduling horizon is 10 hours broken into twenty 30-minute-long time segments.

Table 1 provides a list of the tasks, their duration (in multiples of 30-minute slots), time-window(s), any precedence constraints, and the battery consumption rates. The type of tasks chosen are based on actual tasks performed by the Phoenix Mars lander, but the detailed values are fabricated. The durations and time-windows were chosen with three objectives in mind: to provide an interesting scheduling problem, to have it small enough to fit on the quantum hardware, and to still be an abstracted version of the real Mars lander problem with reasonable values. The consumption rate is the total power consumed every 30 minutes. This consumption is assumed to be constant throughout the duration of the task. Therefore, a baking task that lasts for two hours will consume a total of $0.115 \times 4 = 0.46$ units of power. Consumption rates were chosen so that the total battery power required to complete all tasks sum to 1.

Finally, Table 2 presents an example of the battery charge increase due to solar power during each time segment. We chose these parameters to mimic changing conditions in the course of a day, and to sum to 1 so the production total is equivalent to the consumption total. The time periods at the start and end of the schedule have charging rate of 0, represent times when the sun is not visible to the solar panels. As the day progresses, the sunlight increases until midday, after which the sunlight decreases.

The instances we consider all have the same tasks and task constraints. We vary instances by altering the parame-

ID	Description	Duration	Time-Window(s)	Precedences	Battery Consumption Rate
1	Take Panoramic Picture	2	[6, 16]	-	0.04
2	Measure Weather	1	[2, 8]	-	0.03
3	Take Workspace Picture	3	[0, 13]	-	0.05
4	Gather Soil	3	[3, 16]	3	0.08
5	Bake Sample	4	[6, 20]	4	0.115
6	Send Data	1	[3, 5], [14, 16]	-	0.04

Table 1: Scheduling information regarding tasks.

Time	0 - 4	5	6	7	8	9	10	11	12	13 - 19
Production Rate	0.00	0.03	0.06	0.12	0.15	0.15	0.12	0.06	0.03	0.00

Table 2: Example solar power production rate.

ters that correspond to the battery constraints. The battery has a starting charge that varies during a mission based on the battery consumption and production from previous time periods. For our experiments, we use initial battery levels of either 0.3, 0.5, or 0.7. The maximum capacity changes due to battery degradation over the length of a mission. Early on, the battery capacity may be large, but over time, the battery will be expected to hold much less of a charge. For our experiments, we use battery capacities of either 0.5, 0.7, or 0.9. Finally, we vary the solar power production intensity. Table 2 acts as the baseline power production from the solar panels. We test three power production scenarios where all production is updated to be 75%, 100% or 125% of the baseline. These intensities represents changes in the distance of Mars to the Sun and the visibility of the Sun due to the Martian weather, dust storms and other such obstructions. Ignoring any cases in which the initial battery power is larger than the maximum capacity and any instances that do not have a feasible solution,[1] we are left with a total of 21 problem instances.

Scheduling problem formulation: Let J be the set of all tasks to schedule. For each task $j \in J$, let p_j be its processing time, and W_j its set of time windows. For each time window $w_i \in W_j$, let $r_{j,i}$ and $d_{j,i}$ be the start and end times of that window. Let $T_j = \{t | \, \forall w_i \in W_j, \, r_{j,i} \leq t \leq d_{j,i} - p_j\}$ be the set of all possible start times of task j. To encode the task scheduling problem, we use a time-indexed formulation: for each task j and time index t, we introduce a binary variable $x_{j,t}$, which is 1 if and only task j starts at time t.

A valid schedule for a Mars lander task scheduling problem is a complete assignment for all $x_{j,t}$ variables satisfying the following constraints: (1) no two tasks overlap; (2) precedence constraints between tasks are satisfied; and (3) battery constraints are satisfied.

Quantum Annealing

Quantum computing enables more efficient solution to certain classes of problems than classical computing (Rieffel and Polak 2011; Chuang 2001). While large-scale universal quantum computers are likely decades away, special purpose quantum computational devices are emerging. The first of such are quantum annealers, special purpose hardware designed to run quantum annealing (Farhi et al. 2000; Das and Chakrabarti 2008; Johnson et al. 2011; Smelyanskiy et al. 2012), a metaheuristic that can make use of certain non-classical effects, such as quantum tunneling and quantum interference (Das and Chakrabarti 2008; Boixo et al. 2014), for computational purposes. For classically-trained computer scientists, quantum annealing is one of the most accessible quantum algorithms because of its close ties to classical optimization algorithms such as simulated annealing (Smelyanskiy et al. 2012; Nishimori, Tsuda, and Knysh 2015) and because the most basic aspects of the algorithm can be captured by a classical cost function and classical parameter setting. Special purpose hardware such quantum annealers mean it is now possible to empirically evaluate heuristic quantum algorithms such as quantum annealing, potentially opening up a broader range of applications for quantum computation.

A quantum annealer is designed to minimize Quadratic Unconstrained Binary Optimization (QUBO) problems, problems of minimizing a cost function 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 $\mathbf{x} \in \{\pm 1\}^n$ is a vector of binary-valued variables. Using a quantum annealer requires mapping the problem of interest to QUBO; the following section contains an example for the discrete portion of the Mars lander problem. The solution space of the original problem is mapped to a subset of bit strings, and the cost function encoded in the coefficients such that the minimum value of $f(\mathbf{s})$ and corresponding optimum $\mathbf{s}$ give the solution to the original problem. Penalty functions are added so that bit strings that do not correspond to valid solutions are penalized.

Figure 2 shows the procedure for solving on the quantum annealer the Mars lander scheduling problem without the battery constraint (MLWoB), a single-machine scheduling (SMS) problem which is the discrete part of our hybrid problem. The first step is to encode our MLWoB scheduling problem in the QUBO form described above. To implement the resulting QUBO on the quantum annealer hardware, an additional embedding step is needed. Variables in

[1] We determined the infeasible problem instances by performing a complete search using our algorithm.

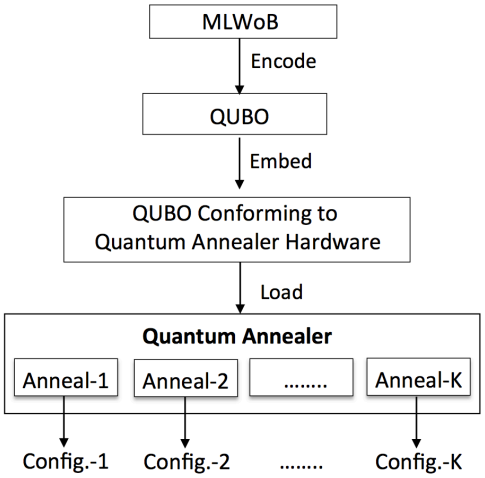

Figure 2: Steps in solving the Mars lander scheduling problem without the battery constraints (MLWoB) using quantum annealer.

the QUBO map to qubits on the hardware. A quadratic constraint can only be implemented directly when the qubits corresponding the variables in the constraint are connected in the hardware. But the physical hardware has only limited connectivity, so often multiple qubits are needed to represent a single variable. Embedding is the process of determining which physical qubits will represent which variables (Kaminsky and Lloyd 2004; Choi 2008; 2011). The coupling strength used between physical qubits representing the same variable is a parameter of the quantum annealing algorithm. More details about embedding and the process of using a quantum annealer to solve application problems can be found in (O'Gorman et al. 2015).

Mapping Mars lander without battery (MLWoB) problems to QUBO: For the rest of this section, we will describe in details how to map MLWoB problems to QUBO form, using the notations, variables, and constraints described in the problem formulation at the end of the previous section.

In the QUBO representation we model constraints in the MLWoB through penalty terms in the cost function objective. To ensure that every task is assigned to exactly one starting time, the term

$$C_{\text{one-start}} = \sum_{j \in J} (\sum_{t \in T_j} x_{j,t} - 1)^2$$

is added to the objective function. To ensure that no two tasks overlap, we introduce another cost function term

$$C_{\text{overlap}} = \sum_{j \in J} \sum_{t \in T_j} \sum_{\substack{l \in J \\ l \neq j}} \sum_{t' \in T'_{j,t} \cap T_l} x_{j,t} x_{l,t'}.$$

The set $T'_{j,t} = \{t' | t \leq t' \leq t + p_j - 1\}$ represents the time points that the Mars lander will be occupied with task j if it starts at time t. Thus, if another task starts during this time, a penalty is incurred. Finally, to model the precedence

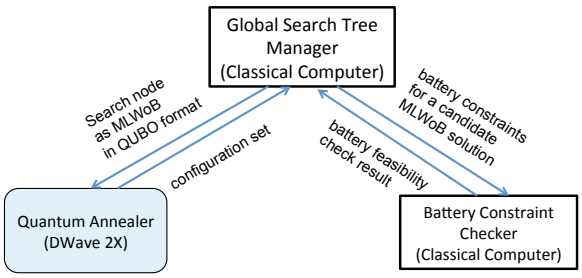

Figure 3: Tree-search based Quantum-Classical Algorithm.

constraints, the cost function term

$$C_{\text{prec}} = \sum_{(j,l) \in P} \sum_{t \in T_l} \sum_{t' \in \tilde{T}_{j,t}} x_{j,t'} x_{l,t}$$

is used. Let P be the set of all pairs of tasks with precedence constraints such that $(j, l) \in P$ implies that task j must be scheduled before task l. The set $\tilde{T}'_{j,t} = \{t' \in T_j | t \leq t'\}$ represents the times where task j cannot start if a task that must start after task j is scheduled at time t. The objective function for the complete QUBO is

$$C = C_{\text{one-start}} + C_{\text{overlap}} + C_{\text{prec}}.$$

A candidate solution will have $C = 0$. If $C > 0$, one or more of the constraints has been violated and the configuration is infeasible.

Quantum Annealing Guided Tree Search

We propose a classical-quantum decomposition algorithm that makes use of (1) a *classical computer* to manage the global search tree and check battery-profile feasibility for schedules produced by the quantum annealer; and (2) a *quantum annealer* to generate the lander schedules, ignoring the battery constraints. Our algorithm is outlined in Figure 3. At a high-level, the main steps are:

1. *Build the global search tree*: with root node representing an empty schedule and leaf nodes representing the complete Mars lander's schedules.
2. *Search node simplification*: Ignore the constraints on the continuous variable representing battery. This results in a MLWoB problem outlined in the previous section.
3. *Run the quantum annealer on the MLWoB problems*: Each run consists of a preset number of anneals, each of which returns a qubit configuration for the MLWoB problem.
4. *Use results from the quantum annealer to guide the high-level tree search*: Each configuration $\mathbf{x}$ returned by the quantum annealer has an associated "energy" value or cost $C(\mathbf{x})$. The results returned from the quantum annealer are used to (1) prune infeasible nodes from the high-level search tree; and (2) guide the subsequent exploration of the remaining search tree.

Checking Battery Constraint: Candidate solutions $\mathbf{x}$ returned by the quantum annealer that have zero cost ($C(\mathbf{x}) =$

Algorithm 1 Quantum Annealing Guided Tree Search.

```
open_nodes: a priority queue
push root_node to open_nodes
while open_nodes ≠ NULL do
    pop first node n from open_nodes
    run quantum annealer on n with pre-defined number
    of anneals to return a set S of configurations
    if any C ∈ S is a battery-ignorant schedule then
        check battery constraints on C
    else
        build partial tree from the configurations
        generate open node(s)
        for each open node generated do
            if node is infeasible then
                prune node
            else
                perform forward checking on node
                push node onto open_nodes
return infeasible
```

0) must be checked to see if they satisfy the battery constraints and are therefore a global solution. The check calculates the battery state at every time point t,

$$B_t = \min(B_{\max}, B_{t-1} + b_t^+ - b_t^-),$$

where b_t^+ is the battery power produced by the solar panels and b_t^- is the battery consumption for the task processed at time t. If $B_t < 0$ at any time, the schedule is infeasible.

Complete Tree Search guided by the Quantum Annealer's Results Profile

For the rest of this section, we will describe in details the guided tree search, which ties the quantum annealer and the classical computation together.

As a stochastic solver, the quantum annealer returns multiple configurations of varying quality, and has no guarantee of finding one with $C(\mathbf{x}) = 0$ when one exists. In order to guarantee that a candidate solution is found when one exists, a systematic search over the state space of the MLWoB problem must be performed. We introduce a binary tree search that is guided by the quantum annealer. We provide pseudocode for the algorithm, and then describe each step in more detail in the subsequent subsections.

Using the Quantum Annealer to build the partial tree. For each job submitted to the quantum annealer, K anneals are performed (see the last step in Figure 2). Of the K configurations returned, $\bar{K}$ are unique with $\bar{K} \leq K$. Configurations are of varying quality with associated cost $C(\mathbf{x}) \geq 0$. All configurations $\mathbf{x}$ with $C(\mathbf{x}) = 0$, representing candidate solutions, are checked to see if the battery constraints are satisfied. If none of them satisfies the battery constraint, the search is continued.

From the $\bar{K}$ configurations, a partial tree can be built, a binary tree with a fixed variable ordering, where variables pertaining to the same task are grouped together. The unshaded nodes (1), (2), and (3) in Figure 4 presents an example of such a partial tree where the square nodes are the configurations. The left (right) branch of a node represents the corresponding variable being set to 0 (1). A problem with N variables will have a tree of size $2^{N+1} - 1$, of which 2^N are leaf nodes corresponding to configurations (assignment of 0 or 1 to all variables). The partial tree is traversed to generate all open nodes (the shaded nodes (A), (B), and (C) in Figure 4), which are defined as nodes that do not exist in the partial tree, but have a sister node that does.

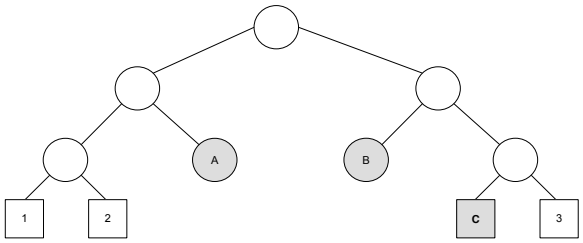

Figure 4: Partial binary search tree example with open nodes shown as shaded nodes. Square nodes here represent the configurations.

Node pruning. Open nodes are pruned by an inference algorithm based on the MLWoB problem constraints and the battery constraints. At any open node, a subset of decisions variables, $x_{j,t}$, have been set. Based on this partial configuration, it is possible to check whether any of the constraints have been violated by looking at the three MLWoB constraints: (1) a task must be assigned to one start time ($\sum_{t \in T_j} x_{j,t} = 1$), (2) no tasks can overlap ($\sum_{j \in J} x_{j,t} \leq 1$), and (3) precedence constraints cannot be violated ($tx_{j,t} + p_j \leq t' x_{l,t'}$). These constraints are checked only for the tasks that have been assigned a start time (or have been assigned 0 to all possible start time variables) at the open node in question. If any of the constraints are violated, the node can be pruned.

Constraint propagation is also performed on the battery constraint. Let $J' \in J$ be the set of already scheduled tasks and y_l their scheduled start times. For $j \in J \setminus J'$, the set of tasks that have yet to be scheduled, consider all remaining decision variables $x_{j,t}$ and check whether

$$y_l + p_l \leq t \text{ or } t + p_j \leq y_l \qquad \forall l \in J'.$$

For all such times t that are still available for task j, the battery check is called for the set of tasks J' starting at times y_l and the task j assuming that it is assigned to start at time t. Let τ_j be the set possible values for t where both the temporal and battery constraints for a task j are satisfied given the partial schedule. If any remaining task to schedule has $\tau_j = \emptyset$, it is impossible to schedule the task with the partial schedule and so the node can be pruned.

If no constraints have been violated, then a final forward checking procedure is performed. The depth of a node indicates the task that is currently being considered. In some cases, it is possible to set the next few variables in the tree that pertain to the task at the current depth. For example, if the start time of a task has already been decided, the remaining start time decision variables for that task can be set to 0.

The current open node can then skip down to the proper location and ignore branches to the right since configurations built from the partial solution of those nodes would violate the single start time constraint.

Node exploration. Once an open node is selected for further exploration, we set the variables of the corresponding partial configurations and invoke the quantum annealer to find the configurations for the remaining variables. New configurations are found and the partial tree beneath the open node is built in the same manner as described above. Because configurations are built upon the partial configurations of the open node, all configurations found will not have been explored previously.

Node selection. To decide which node to explore next, we considered three node selection heuristics. The first uses task scheduling slackness. The second heuristic uses the cost value $C(\mathbf{x})$ of the configurations returned by the quantum annealer to guide selection. The final combines slackness with the cost value. We call the first heuristic *Slack-Only*, the second *QA-Only*, and the last *Weighted*.

Slack-Only measures the remaining slack of times left that a task can be scheduled as $s_j = |\tau_j|$, where τ_j is the set defined during constraint propagation. We calculate a scoring function, $S = \left(\prod_{j \in J \setminus J'} s_j\right)^{\frac{1}{|J \setminus J'|}}$, for each node. The geometric mean is taken over the slack of each remaining task to better compare nodes at different depths and to avoid partial candidate solutions that greatly restrict some tasks even if others have large slack. Nodes with more slack have more time in which to fit all the remaining tasks, so slack is a reasonable measure of how likely candidate solutions $\mathbf{x}$ with $C(\mathbf{x}) = 0$ exists below the node.

QA-Only uses the cost value $C(\mathbf{x})$ of configurations to guide node selection. A score C^* is defined for each open node as the lowest $C(\mathbf{x})$ for any configuration $\mathbf{x}$ that has been found from the sub-tree of the open node's parent. This is a proxy measure for how good the partial configuration is for the MLWoB. Here, we choose the open node with the minimum cost value since these configurations are closer in cost to candidate solutions.

Weighted (α) weights the Slack-Only and QA-Only heuristics. The scoring function that Weighted (α) uses is $(1 - \alpha)S - \alpha C^*$, where $\alpha \in [0, 1]$ is a control parameter that determines how much influence the cost function will have on the scoring function. In general, we see the slack function ranges between 1 and 7. The cost value has a much greater range, but the nodes of interest will have low cost within the same range as the slack since these solutions are closer values to candidate solutions. Note that we subtract the cost value from the scoring function as a lower cost value is associated with a better configuration. If $\alpha = 1$, then the scoring function is the QA-Only heuristic and if $\alpha = 0$ then the scoring function will be equivalent to the Slack-Only heuristic. Thus, an open node that has a parent that was found to be part of a good MLWoB schedule will have increased priority when $\alpha > 0$. The intuition for this heuristic is that it is better to explore parts of the tree where other good configurations were found rather than parts where only poor configurations have been found.

Conditions for Termination. The algorithm continues to explore open nodes until either a global solution has been found or there are no longer any open nodes to explore. An empty open node list implies that no global solution exists.

Empirical Evaluation and Analysis

We evaluated our algorithm 10 times on each of the 21 Mars lander instances. Since the MLWoB problem is the same across instances, the initial QUBO at the root node is always the same. This QUBO has 52 variables, one for each possible start time of a task. For example, the "measure weather" action has time-window [2, 8] and a processing time of 1, so it will have 6 binary variables representing the possible start times of the task from time 2 until time 7. Embedding this QUBO into the D-Wave 2x hardware results in the use of many more qubits. For our experiments, we generated a single embedding at the root node and used the same embedding for all problems. This embedding has 764 qubits. Nodes deeper in the tree correspond to restricted MLWoB problems where some variables are already set, so we can use the same embedding but with a subset of the variables.

We examine four variants of our algorithm:

1. Slack-Only
2. QA-Only
3. Weighted heuristic with four different weightings
4. a *Random-Sample* baseline algorithm that does not make use of the quantum annealer, but instead generates configurations from open nodes through random sampling.

The baseline algorithm naively explores the tree without directly trying to find feasible MLWoB schedules. Constraint propagation is still performed on the nodes of the tree to guide search and the node selection is done by choosing the node with the largest geometric mean of the slack.

We implement all classical components of our algorithms in Python and run quantum annealing on the D-Wave 2X machine housed at NASA Ames. To explore an open node, we submit a job request with $K = 10,000$ anneals 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) with one exception. Our previous experiments showed that for the MLWoB problem, increasing by 5 times the default D-Wave parameter for the coupling strength between qubits representing the same single variable resulted in greatly improved performance. For Random-Sample, we generate 10,000 samples at each explored node.

Results: Table 3 displays the performance of the different variants. The number of explored nodes correlates with the effort required to solve a problem. The number of unique configurations found defines the size of the tree built. A larger tree results in a larger computation on the classical computer because the number of open nodes increases.

	avg. # of open nodes explored	avg. # of configurations found
Random-Sample	420.27	4,088,269.03
Slack-Only (0.0	4.10	2,143.21
Weighted (0.2)	3.27	2,042.94
Weighted (0.4)	2.38	1,795.79
Weighted(0.6)	2.27	1,784.30
Weighted (0.8)	2.22	1,728.33
QA-Only (1.0)	3.25	1,822.60

Table 3: Results for the algorithm variants on the twenty-one problem instances considered: solving each instance ten times for each variant. Slack-Only uses the quantum annealing configurations to build the tree and Weighted and QA-Only goes a step further to use the objective function from configurations to guide node selection, with the Weighted (0.8) variant performing best.

The results show a reduction in the number of open nodes explored with guidance from the configurations found by the quantum annealer. The most naive approach, Random-Sample, explores the most open nodes. Slack-Only and QA-Only significantly reduce the search effort as the configurations used to build the partial tree is guided by a better solver. All values of α tested for the Weighted variation further improve the results. A deeper understanding of how to choose the α value is left for future work.

Not only is exploration happening in a more directed manner in our algorithms compared to Random-Sample, but the construction of the tree is more focused. For Random-Sample, a much larger portion of the tree is built during exploration than for the quantum annealing guided searches; Random-Sample usually returns $10,000$ distinct configurations, whereas the quantum annealer often returns the same configuration multiple times as it tries to find good quality MLWoB schedules. Random-Sample also had significantly more open nodes due to the larger partial trees built.

Figure 5 provides further insight into the performance of the different variants. We see that the Weighted variants generally find global solutions much faster than the Slack-Only or QA-Only variants. The Weighted variants exhibit more regular distributions. QA-Only is able to solve many problems quickly, but has a heavy tail as strictly using low cost value configurations to guide search can lead to the search staying longer in bad branches. Slack-Only is not able to quickly find global solutions, but does solve many problems at the sixth open node explored. We suspect this behavior is in part due to the order of the variables, with variables representing a single job's start time clustered.

Figure 6 indicates how often a node of a certain depth is explored for all variants other than Random-Sample. We see that Slack-Only spends proportionately more time at shallower depths. Particularly, between depths of 10 and 26, we see a large increase in exploration. Using the quality of configurations from the quantum annealer, search is focused towards deeper nodes in the tree. Weighted (0.6 and 0.8) along with QA-Only are the only variants that searches nodes deeper than a depth of 26, with Weighted (0.8) and QA-Only searching even to depth of 33. As more emphasis is given to the $C(x)$ values to guide node selection, the search algorithm will spend time at deeper nodes near what it believes are other good configurations.

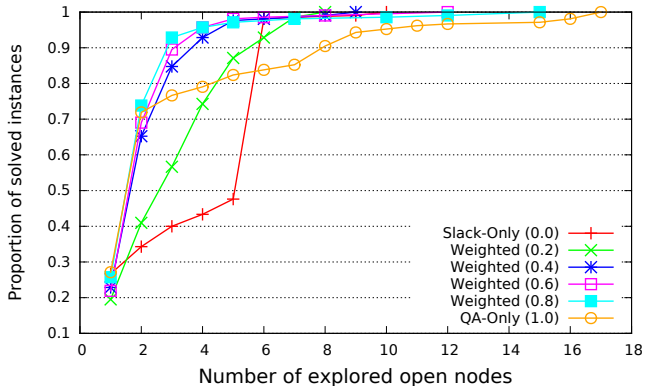

Figure 5: Cumulative proportion of instances solved versus the number of open nodes explored. This graph provides a better picture into the distribution of performance over the set of instances tested. Random-Sample results are omitted.

The results presented should not to be considered thorough benchmarks of the quantum annealer's performance. Several performance boosting methodologies have been developed (Perdomo-Ortiz et al. 2015; Venturelli et al. 2015; Vinci et al. 2015) that can improve the quality of the configurations by orders of magnitude with respect to the basic use of the annealer. In order to keep the discussion restricted to the topic of the decomposition approach, other than for the coupling strength between qubits representing the same logical variables, we used the default parameters of the D-Wave 2X APIs. To make the best use of quantum annealers, it will be crucial to take full advantage of state-of-the-art tuning and programming techniques.

Related Work

Given the relative novelty of the quantum annealing hardware, research in this area has been limited. Pure quantum annealing formulations have been built for some planning and scheduling problems (Rieffel et al. 2015; Venturelli, Marchand, and Rojo 2015). 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 in deep learning, quite a different use of sampling than 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

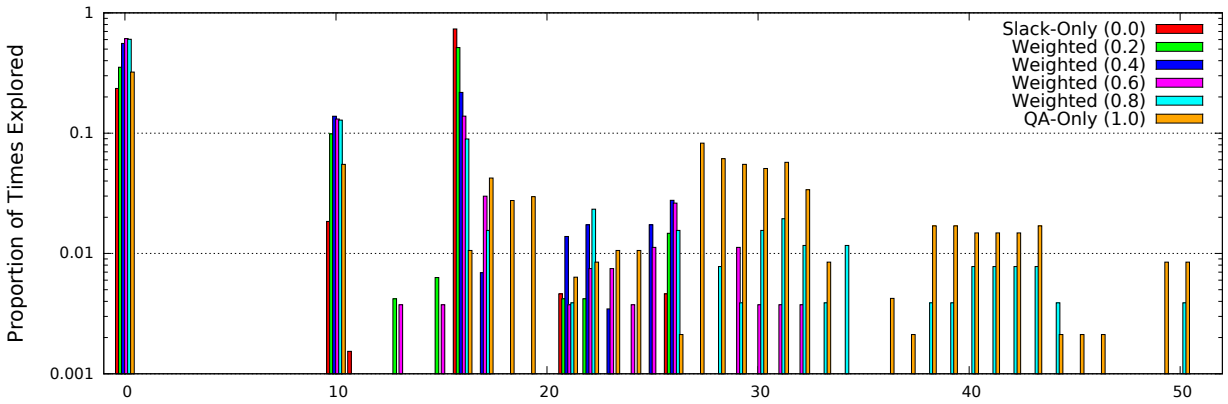

Figure 6: Node Depth Histogram. Proportion of time spent exploring nodes of various depths. Using the quantum annealing results guides search towards deeper nodes that were found to be near other good configurations.

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. Furthermore, 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 all results from a quantum annealer are used to prune and guide the search. We tested multiple heuristics in this framework on instances of a Mars Lander activity scheduling problem. The framework enables the use of a stochastic quantum-annealing solver within a complete search framework. In summary, key observations include (1) results from the quantum annealer can effectively prune and guide the search process; (2) taking into account all, and not just the low-cost, configurations returned by the quantum annealer is useful. The extent to which quantum annealing results are used in node selection improves performance, suggesting that further exploration of node selection metrics incorporating quantum annealing results is warranted. Other metrics of different forms should be explored. For example, such a metric could incorporate other information, such as how close the configuration is to being feasible in terms of battery power.

An obvious direction for future work is to apply this framework to a larger variety of problems beyond the Mars Lander task scheduling problem. Our approach is adaptable to a large set of problems with a variety of characteristics, which can be incorporated directly into the QUBO or allocated to a classical sub-solver in a similar manner to how we handle the battery constraint. A particularly interesting set to explore would be problems with more complicated battery models, such as models in which production and consumption of battery power is nonlinear and depends on the battery's state. Also, the framework could be extended to larger problems in which the battery-ignorant problem doesn't fit on the D-Wave machine, but parts of the tree are explored via classical and quantum guided tree search.

Our framework is not limited to strictly quantum-classical algorithms. The tree search we propose can just as easily be applied to classical-classical decompositions. In particular, it allows the use of specialized heuristic solvers for difficult problems that can be decomposed in a way similar to the Mars lander task scheduling problem. The heuristic can be used to find good solutions quickly, while the framework ensures that a complete search is being performed.

Another potential extension of this work is to incorporate further classical heuristics into the tree search, borrowing ideas from the extensive classical literature, such as variable ordering (Smith 1996), conflict analysis and cutting planes (Achterberg 2007; Kelley 1960), and discrepancy-based search (Harvey and Ginsberg 1995; Walsh 1997). 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 variable would need to be added to the QUBO formulation in order to incorporate constraints from cutting planes or nogood cuts. For this reason, it is interesting to explore the design of such extensions while keeping additional variables to a minimum.

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

Acknowledgement: 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*.

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. 2008. Minor-embedding in adiabatic quantum computation: I. the parameter setting problem. *Quantum Information Processing* 7(5):193–209.

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

Chuang, M. N. L. 2001. *Quantum Computing and Quantum Information*. Cambridge: Cambridge University Press.

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.

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.

Johnson, M. W.; Amin, M. H. S.; Gildert, S.; and et al. 2011. Quantum annealing with manufactured spins. *Nature* 473:194–198.

Kaminsky, W., and Lloyd, S. 2004. Scalable architecture for adiabatic quantum computing of NP-hard problems. *Quantum Computing and Quantum Bits in Mesoscopic Systems* 229–236.

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

Nishimori, H.; Tsuda, J.; and Knysh, S. 2015. Comparative study of the performance of quantum annealing and simulated annealing. *Phys. Rev. E* 91:012104.

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.

Perdomo-Ortiz, A.; Fluegemann, J.; Biswas, R.; and Smelyanskiy, V. N. 2015. A performance estimator for quantum annealers: Gauge selection and parameter setting. *arXiv preprint arXiv:1503.01083*.

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.

Smith, B. M. 1996. Succeed-first or fail-first: A case study in variable and value ordering. In *Proceedings of the ILOG Solver and ILOG Scheduler Second International Users' Conference*, 321–330.

Venturelli, D.; Mandrà, S.; Knysh, S.; O'Gorman, B.; Biswas, R.; and Smelyanskiy, V. 2015. Quantum optimization of fully connected spin glasses. *Physical Review X* 5(3):031040.

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

Vinci, W.; Albash, T.; Paz-Silva, G.; Hen, I.; and Lidar, D. A. 2015. Quantum annealing correction with minor embedding. *Physical Review A* 92(4):042310.

Walsh, T. 1997. Depth-bounded discrepancy search. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI)*, volume 97, 1388–1393.

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.