QPU Access Time and Billing Estimates for Decomposition Strategies

Based on D-Wave Operation and Timing Documentation

Agricultural Land Allocation Optimization Project
OQI-UC002-DWave

November 26, 2025

Abstract

This report provides detailed estimates of Quantum Processing Unit (QPU) access time and associated billing charges for various quantum-classical hybrid decomposition strategies implemented in the agricultural land allocation optimization project. Estimates are derived from official D-Wave documentation on operation timing and validated against actual solver parameters used in the codebase. We analyze four QPU-enabled approaches: `current_hybrid`, `benders_qpu`, `dantzig_wolfe_qpu`, and `admm_qpu`, providing timing breakdowns for problem sizes ranging from 10 to 50 farm units.

Contents

1 Introduction **3**

2 D-Wave Timing Model **3**
- 2.1 QPU Access Time Components . . . 3
- 2.2 Typical Parameter Values . . . 3
- 2.3 Readout Time Dependency on Problem Size . . . 4

3 Decomposition Strategy Analysis **4**
- 3.1 Current Hybrid Strategy . . . 4
- 3.2 Benders QPU Strategy . . . 4
- 3.3 Dantzig-Wolfe QPU Strategy . . . 5
- 3.4 ADMM QPU Strategy . . . 5

4 QPU Access Time Estimates **5**
- 4.1 Direct QPU Access (DWaveSampler) . . . 5
  - 4.1.1 Calculation Methodology . . . 5
  - 4.1.2 Problem Size Estimates . . . 6
- 4.2 Hybrid Solver Access (LeapHybrid) . . . 6

5 Total Billed Time by Strategy **6**
- 5.1 Summary Table . . . 7
- 5.2 Detailed Breakdown: 10 Farms Configuration . . . 7
- 5.3 Detailed Breakdown: 25 Farms Configuration . . . 7
- 5.4 Detailed Breakdown: 50 Farms Configuration . . . 7

6 Billing and Quota Consumption **7**
- 6.1 D-Wave Billing Model . . . 7
- 6.2 Monthly Quota Estimates . . . 8

7 Optimization Recommendations **8**
7.1 Reducing QPU Access Time 8
7.2 Optimized Parameter Set 8

8 Conclusions **8**
8.1 Key Findings 8
8.2 Recommended Strategy Selection 9

A SAPI Timing Fields Reference **9**

B Example SAPI Timing Response **9**

1 Introduction

The D-Wave quantum computer operates through Quantum Machine Instructions (QMIs), each requiring a sequence of operations with well-defined timing characteristics. Understanding these timings is critical for:

- Estimating computational costs before execution
- Optimizing solver parameters for efficiency
- Comparing quantum and classical approaches fairly
- Budget planning for QPU quota consumption

This report focuses on the **QPU access time**, which represents the actual time billed by D-Wave and is computed as:

$$T = T_p + \Delta + T_s \tag{1}$$

where:

- T_p = Programming time (typically 10–20 ms for Advantage systems)
- Δ = Overhead time for low-level operations (10–20 ms)
- T_s = Total sampling time

2 D-Wave Timing Model

2.1 QPU Access Time Components

According to official D-Wave documentation, the sampling time is computed as:

$$\frac{T_s}{R} \approx T_a + T_r + T_d \tag{2}$$

where:

- R = Number of reads (samples)
- T_a = Anneal time per sample
- T_r = Readout time per sample
- T_d = Delay time per sample

The delay time consists of optional components:

$$T_d = \text{readout\_thermalization} + \text{reduce\_intersample\_correlation} + \text{reinitialize\_state} \tag{3}$$

2.2 Typical Parameter Values

Table 1 shows the default parameter values used in our implementation.

Table 1: Default D-Wave Solver Parameters (from `dwave_qpu_adapter.py`)

Parameter	**Value**	**Unit**
`num_reads`	1000	samples
`annealing_time` (T_a)	20.0	µs
`programming_thermalization`	1000.0	µs
`readout_thermalization`	1000.0	µs
`reduce_intersample_correlation`	true	–
`reinitialize_state`	true	–

2.3 Readout Time Dependency on Problem Size

A critical insight from the documentation is that **readout time depends on the number of qubits**:

- Small problems (∼12 qubits): $T_r \approx 25\,\mu s$
- Large problems (∼5000 qubits): $T_r \approx 150\,\mu s$

For our agricultural optimization problems with n farms and 27 foods, the number of binary variables is:

$$N_{\text{vars}} = n \times 27 \tag{4}$$

With minor embedding overhead (chain factor ≈ 2–4), the estimated qubit count becomes:

$$N_{\text{qubits}} \approx (2 \text{ to } 4) \times N_{\text{vars}} \tag{5}$$

3 Decomposition Strategy Analysis

Our codebase implements seven decomposition strategies, of which four use QPU:

1. `current_hybrid`: Uses LeapHybridCQMSampler or LeapHybridBQMSampler
2. `benders_qpu`: Benders decomposition with QPU for master problem
3. `dantzig_wolfe_qpu`: Column generation with QPU for pricing
4. `admm_qpu`: ADMM with QPU for Y-subproblem

3.1 Current Hybrid Strategy

The `current_hybrid` strategy uses D-Wave's Leap Hybrid solvers, which abstract away direct QPU management.

Table 2: Current Hybrid Strategy Timing Characteristics

Characteristic	**Value**
Solver Type	LeapHybridCQMSampler / LeapHybridBQMSampler
Minimum Time Limit	3.0 seconds
QPU Calls per Solve	Variable (managed internally)
Typical QPU Access Time	10–100 ms per hybrid call
QPU Time Fraction	≈ 1–3% of total solve time

Billing Model: Hybrid solvers bill based on problem complexity and actual QPU access within the hybrid workflow. The `qpu_access_time` field in the response indicates actual QPU time consumed.

3.2 Benders QPU Strategy

The `benders_qpu` strategy uses QPU to solve the master problem (binary Y variables) in each Benders iteration.

Table 3: Benders QPU Strategy Timing Characteristics

Characteristic	Value
Solver Type	LeapHybridBQMSampler (or SimulatedAnnealing fallback)
Max Iterations	50
QPU Calls per Solve	Up to 49 (first iteration classical)
Problem per QPU Call	$n_{\text{farms}} \times n_{\text{foods}}$ binary variables
Hybrid Time Limit	3.0 seconds per call

3.3 Dantzig-Wolfe QPU Strategy

The `dantzig_wolfe_qpu` strategy uses QPU for the pricing subproblem in column generation.

Table 4: Dantzig-Wolfe QPU Strategy Timing Characteristics

Characteristic	Value
Solver Type	LeapHybridBQMSampler (or SimulatedAnnealing fallback)
Max Iterations	50
QPU Calls per Solve	Up to 50
Problem per QPU Call	Pricing subproblem (smaller than full problem)
Hybrid Time Limit	3.0 seconds per call

3.4 ADMM QPU Strategy

The `admm_qpu` strategy uses QPU for the Y-subproblem in each ADMM iteration.

Table 5: ADMM QPU Strategy Timing Characteristics

Characteristic	Value
Solver Type	LeapHybridBQMSampler (or SimulatedAnnealing fallback)
Max Iterations	10 (default)
QPU Calls per Solve	Up to 10
Problem per QPU Call	$n_{\text{farms}} \times n_{\text{foods}}$ binary variables
Hybrid Time Limit	3.0 seconds per call

4 QPU Access Time Estimates

4.1 Direct QPU Access (DWaveSampler)

For strategies using direct QPU access (not hybrid), we can estimate the QPU access time using the formulas from Section 2.

4.1.1 Calculation Methodology

Using the timing model from D-Wave documentation:

$$\begin{aligned} T_p &= \text{typical\_programming\_time} + \text{programming\_thermalization} && (6) \\ &\approx 15\,\text{ms} + 1\,\text{ms} = 16\,\text{ms} && (7) \end{aligned}$$

For sampling time per read:

$$
\begin{aligned}
T_{\text{sample}} &= T_a + T_r + T_d && (8)\\
T_a &= 20\,\mu\text{s (default annealing time)} && (9)\\
T_r &= 25\,\mu\text{s to }150\,\mu\text{s (depends on qubits)} && (10)\\
T_d &\approx 21\,\mu\text{s} + \text{readout\_thermalization} && (11)
\end{aligned}
$$

With `readout_thermalization` $= 1000\,\mu$s:

$$T_{\text{sample}} \approx 20 + T_r + 1021 = 1041 + T_r \ (\mu\text{s}) \tag{12}$$

4.1.2 Problem Size Estimates

Table 6 shows QPU access time estimates for different problem sizes.

Table 6: Direct QPU Access Time Estimates (1000 reads)[1]

Farms	Variables	Est. Qubits	T_r (μs)	T_s (ms)	T (ms)	Cost
10	270	∼800	60	112.1	143	0.14 min
25	675	∼2000	90	115.1	146	0.15 min
50	1350	∼4000	130	119.1	150	0.15 min
100	2700	∼5000+	150	121.1	152	0.15 min

4.2 Hybrid Solver Access (LeapHybrid)

For hybrid solvers, the timing model is different:

- Minimum run time: 3 seconds
- QPU access time is a fraction of total run time
- Typical QPU access: 10–100 ms per hybrid call

Table 7: Estimated QPU Time per Hybrid Solver Call

Farms	Problem Size	Hybrid Time (s)	Est. QPU Time (ms)
10	Small	3.0	20–40
25	Medium	3.0–5.0	40–80
50	Large	5.0–10.0	80–150
100	Very Large	10.0–30.0	100–300

5 Total Billed Time by Strategy

Based on the analysis above, we estimate the total QPU access time (billed time) for each strategy.

[1] Cost based on quota conversion rate of 1 and 1 minute = 60,000 ms

5.1 Summary Table

Table 8: Total Estimated QPU Access Time by Strategy (25 farms, 27 foods)[2]

Strategy	QPU Calls	Time/Call (ms)	Total QPU (ms)	Total QPU (s)	Quota (min)
current_hybrid	1	40–80	40–80	0.04–0.08	$<$0.01
benders_qpu	10–30	40–80	400–2400	0.4–2.4	0.01–0.04
dantzig_wolfe_qpu	10–30	30–60	300–1800	0.3–1.8	0.01–0.03
admm_qpu	10	40–80	400–800	0.4–0.8	0.01–0.01

5.2 Detailed Breakdown: 10 Farms Configuration

Table 9: QPU Timing Breakdown for 10-Farm Configuration

Strategy	Iterations	QPU/Iter (ms)	Total QPU (ms)	Total Hybrid (s)
current_hybrid	1	20–40	20–40	3.0
benders_qpu	∼15	20–40	300–600	45–75
dantzig_wolfe_qpu	∼20	15–30	300–600	60–100
admm_qpu	10	20–40	200–400	30–40

5.3 Detailed Breakdown: 25 Farms Configuration

Table 10: QPU Timing Breakdown for 25-Farm Configuration

Strategy	Iterations	QPU/Iter (ms)	Total QPU (ms)	Total Hybrid (s)
current_hybrid	1	40–80	40–80	3.0–5.0
benders_qpu	∼20	40–80	800–1600	60–120
dantzig_wolfe_qpu	∼25	30–60	750–1500	75–150
admm_qpu	10	40–80	400–800	30–50

5.4 Detailed Breakdown: 50 Farms Configuration

Table 11: QPU Timing Breakdown for 50-Farm Configuration

Strategy	Iterations	QPU/Iter (ms)	Total QPU (ms)	Total Hybrid (s)
current_hybrid	1	80–150	80–150	5.0–10.0
benders_qpu	∼25	80–150	2000–3750	75–180
dantzig_wolfe_qpu	∼30	60–120	1800–3600	90–210
admm_qpu	10	80–150	800–1500	30–60

6 Billing and Quota Consumption

6.1 D-Wave Billing Model

D-Wave charges based on `qpu_access_time`, with rates depending on the solver:

[2]Actual iterations depend on convergence

- **Direct QPU (DWaveSampler)**: Full QPU access time is billed
- **Hybrid Solvers**: Only actual QPU access within hybrid is billed
- **Quota Conversion Rate**: Typically 1 (1 ms QPU time = 1 ms quota)

6.2 Monthly Quota Estimates

Assuming monthly free quota of 1 minute (60,000 ms) for Leap accounts:

Table 12: Estimated Runs per Month (1 minute free quota)

Strategy	QPU/Run (ms)	Runs/Month	Sufficient?
`current_hybrid`	60	1000	✓ Excellent
`benders_qpu` (10 farms)	450	133	✓ Good
`benders_qpu` (25 farms)	1200	50	✓ Adequate
`benders_qpu` (50 farms)	2875	20	∼ Limited
`admm_qpu` (25 farms)	600	100	✓ Good

7 Optimization Recommendations

7.1 Reducing QPU Access Time

Based on the timing model, we can optimize QPU usage:

1. **Reduce `num_reads`**: Current default is 1000; for iterative methods, 100–500 may suffice
2. **Reduce `readout_thermalization`**: Current 1000 µs can be reduced to 0 µs for faster sampling
3. **Use shorter annealing times**: For quick exploration, 5 µs to 10 µs may work
4. **Limit iterations**: Early stopping when convergence is achieved

7.2 Optimized Parameter Set

Table 13: Optimized vs Default Parameters for Iterative Methods

Parameter	Default	Optimized
`num_reads`	1000	200
`annealing_time`	20 µs	10 µs
`readout_thermalization`	1000 µs	0 µs
Estimated QPU/call	146 ms	23 ms
Speedup Factor	1×	6×

8 Conclusions

8.1 Key Findings

1. **Hybrid solvers are most efficient**: The `current_hybrid` approach provides the best QPU time efficiency, as the hybrid solver optimizes QPU usage internally.

2. **Iterative methods accumulate QPU time**: Strategies like `benders_qpu` and `dantzig_wolfe_qpu` can consume 10–50× more QPU time than single-call hybrid approaches due to multiple iterations.

3. **ADMM is most predictable**: With a fixed 10 iterations (default), `admm_qpu` provides consistent and predictable QPU consumption.

4. **Problem size impact is moderate**: QPU access time scales sublinearly with problem size due to the dominance of programming and thermalization overhead.

5. **Free quota is adequate for development**: The standard 1 minute/month Leap quota supports 50–1000 optimization runs depending on strategy and problem size.

8.2 Recommended Strategy Selection

Table 14: Strategy Recommendations by Use Case

Use Case	Recommended Strategy	Rationale
Production/Many runs	`current_hybrid`	Lowest QPU cost
Research/Best quality	`benders_qpu`	Best solution quality
Quick prototyping	`admm_qpu`	Predictable time
Large problems	`current_hybrid`	Scales best

A SAPI Timing Fields Reference

The following fields are returned by the D-Wave SAPI for timing analysis:

Table 15: SAPI Timing Fields

Field	Description
`qpu_access_time`	Total time in QPU (billed time)
`qpu_programming_time`	Time to program the QPU (T_p)
`qpu_sampling_time`	Total time for all samples (T_s)
`qpu_anneal_time_per_sample`	Time for one anneal (T_a)
`qpu_readout_time_per_sample`	Time for one read (T_r)
`qpu_delay_time_per_sample`	Delay between anneals (T_d)
`qpu_access_overhead_time`	Overhead time (Δ)
`total_post_processing_time`	Server-side postprocessing

B Example SAPI Timing Response

From D-Wave documentation:

```
{'qpu_sampling_time': 80.78,
 'qpu_anneal_time_per_sample': 20.0,
 'qpu_readout_time_per_sample': 39.76,
 'qpu_access_time': 16016.18,
 'qpu_access_overhead_time': 10426.82,
 'qpu_programming_time': 15935.4,
 'qpu_delay_time_per_sample': 21.02,
 'total_post_processing_time': 809.0,
```

```
'post_processing_overhead_time': 809.0}
```

This example shows:

- Programming time dominates (∼16 ms)
- Sampling is very fast (∼81 µs for the entire run)
- Total QPU access time: ∼16 ms