KOREAN MANUSCRIPT

강건한 QAOA 파라미터 선택에서 나타나는 시뮬레이터-하드웨어 전이 간극

한글 원고 초안 · Generated from paper/full-draft-ko.md

강건한 QAOA 파라미터 선택에서 나타나는 시뮬레이터-하드웨어 전이 간극

초안 날짜: 2026-05-26

초록

QAOA(Quantum Approximate Optimization Algorithm)는 보통 선택된 파라미터가 달성한 목적함수의 기댓값으로 평가된다. 하지만 근미래 양자 장치에서는 시뮬레이션에서 잘 동작한 파라미터가 실제 장치에서도 그대로 practical winner가 된다고 보기 어렵다. 실제 장치에는 라우팅, calibration drift, readout error, 유한 샷 샘플링, backend별 노이즈가 함께 작용하기 때문이다. 본 논문은 작은 Max-Cut 문제에서 강건성 인식 QAOA 파라미터 선택기(RA-QAOA)를 분석한다. RA-QAOA는 파라미터를 ideal 또는 noisy expected cut 하나만으로 고르지 않고, noisy expected cut, ideal-to-noisy degradation, optimal-solution probability, shot variance, noise sensitivity를 함께 사용해 후보를 점수화한다.

6, 8, 10개 노드 랜덤 그래프 150개를 사용한 p=2 시뮬레이션 benchmark에서 robust selector는 ideal/noisy selector 대비 noise sensitivity를 평균 0.412928 낮췄고, 95% 신뢰구간은 [-0.494519, -0.331337]이었다. 1-qubit gate, 2-qubit gate, readout error exposure를 근사적으로 반영한 hardware-like proxy에서도 p=2 sensitivity delta -0.946489가 관찰되었다. 다만 이 안정성 이득은 expected cut 비용을 동반한다. 즉 RA-QAOA는 raw-performance optimizer가 아니다.

이후 4-node square Max-Cut, p=2 설정으로 실제 IBM QPU pilot을 수행했다. 비교한 selector family는 `robust`, `noisy_expected`, `optimizer`이다. IBM `ibm_marrakesh` 3개 job과 IBM `ibm_kingston` 1개 job 전체에서 `noisy_expected`가 두 backend 모두에서 평균 QPU expected cut과 success probability 1위였다. 전체 4개 QPU job 평균에서 `noisy_expected`는 QPU E[C] 3.645264, success probability 0.837158을 보였고, robust selector는 QPU E[C] 3.597412, success probability 0.816284를 보였다. 또한 job-time calibration metadata, transpiler layout, active physical qubits, per-qubit readout-mitigation check를 회수했다. Independent readout mitigation은 모든 success probability를 높였지만, 어떤 within-job selector rank도 바꾸지 않았다.

따라서 본 논문의 결론은 RA-QAOA가 하드웨어에서 지배적이라는 것이 아니다. 핵심 결론은 simulated robustness와 real-device performance를 분리해서 보고해야 한다는 것이다. 본 case study에서 robust selector는 시뮬레이션상 stability-performance trade-off를 드러냈고, QPU pilot은 simulator-to-hardware transfer gap을 보여주었다.

1. 서론

QAOA는 조합 최적화 문제를 위한 hybrid variational algorithm이다. 문제 Hamiltonian과 mixer Hamiltonian을 번갈아 적용하고, 고전 optimizer가 layer parameter를 선택한다 **[@farhi2014qaoa]**. Max-Cut 실험에서는 파라미터 선택 결과를 보통 best expected cut value로 요약한다. 이 값은 ideal simulation 또는 특정 noisy model 아래에서 계산된다.

하지만 이 단일 축 평가는 근미래 양자 프로세서에서는 불충분하다. NISQ 장치는 finite sampling, noisy gates, readout error, routing constraints, calibration drift의 영향을 받는다 **[@preskill2018nisq]**. 시뮬레이터에서 최고 expected value를 보인 파라미터가 작은 노이즈 변화에 민감하거나 실제 backend에서 다르게 라우팅될 수 있다. 연구가 highest simulated expected value만 보고하면, 그 결과가 실제 hardware로 얼마나 잘 전달되는지 과장할 위험이 있다.

본 논문은 더 좁고 방어 가능한 질문을 다룬다. 강건성 인식 파라미터 selector가 시뮬레이션 노이즈에 덜 민감한 QAOA 파라미터를 찾을 수 있는가? 그리고 그 강건성이 실제 QPU 실행으로 전달되는가? 우리는 작은 Max-Cut instance에서 이 질문을 분석한다. 제안하는 RA-QAOA는 새로운 QAOA circuit ansatz가 아니다. 후보 파라미터 중 어떤 것을 선택할지 결정하는 selection criterion을 바꾼 것이다.

결과는 두 주장으로 분리된다. 첫째, RA-QAOA는 테스트한 benchmark와 hardware-like proxy에서 simulated noise sensitivity를 낮춘다. 둘째, 이 simulated robustness는 real-device dominance로 자동 전이되지 않는다. 실제 QPU pilot에서는 noisy expected-cut selector가 두 IBM backend 모두에서 practical winner였다.

본 논문의 기여는 다음과 같다.

1. QAOA 파라미터 선택을 expected value, degradation, success probability, shot variance, noise sensitivity를 포함하는 multi-metric selector 문제로 정의한다.
2. 60개 및 150개 그래프 시뮬레이션 benchmark에서 optimizer, multi-start optimizer, ablation, size-stratified, hardware-like proxy 비교를 수행한다.
3. IBM `ibm_marrakesh`와 `ibm_kingston`에서 작은 real-QPU pilot을 수행해 simulated selector ranking과 QPU selector ranking이 달라질 수 있음을 보인다.
4. job-level calibration, recovered layout, active-qubit metadata, independent readout-mitigation check를 포함해 pilot을 auditable하게 만든다.
5. 결과를 broad hardware validation 또는 quantum-advantage claim이 아니라 simulator-to-hardware transfer-gap case study로 위치시킨다.

2. 배경 및 관련 연구

QAOA는 조합 최적화 문제의 근사해를 찾기 위한 variational approach로 제안되었다 **[@farhi2014qaoa]**. QAOA는 parameterized quantum circuit과 classical optimization loop를 결합하는 variational quantum algorithm 계열에 속한다 **[@peruzzo2014vqe]**. Weighted Max-Cut과 관련 graph problem에서는 parameter setting과 parameter transfer가 활발히 연구되고 있다 **[@sureshbabu2023weightedqaoa; @montanez2024transfer]**.

근미래 hardware는 QAOA 결과 해석 방식을 바꾼다. NISQ 실험은 device noise, calibration drift, sampling cost, mitigation assumption의 제약을 받는다 **[@preskill2018nisq]**. 실제 장치 utility 실험도 mitigation과 backend detail이 해석을 크게 좌우할 수 있음을 보여준다 **[@kim2023utility]**. 따라서 hardware result는 routing, calibration, layout, measurement assumption이 보고되지 않는 한 simulated result의 단순한 연장으로 취급해서는 안 된다.

본 연구는 QAOA performance-optimization 연구와 다르다. RA-QAOA가 더 좋은 optimizer라고 주장하지 않는다. 대신 stability-aware selector가 덜 noise-sensitive한 simulated choice를 만들 수 있는지, 그리고 그 choice가 실제 장치에서도 경쟁력을 유지하는지를 묻는다.

3. 방법

3.1 Max-Cut QAOA

그래프 G = (V, E)에 대해 각 bitstring z는 vertex를 두 partition으로 나눈다. Max-Cut 값 C(z)는 partition을 가로지르는 edge의 개수 또는 weight 합이다. p-layer QAOA state는 다음과 같다.

```
|psi(gamma, beta)> =
  product_l exp(-i beta_l H_M) exp(-i gamma_l H_C) |+>^n.
```

p=2에서는 parameter vector가 두 개의 gamma와 두 개의 beta를 포함한다. 본 구현은 tied parameter가 아니라 independent p-layer parameter control을 사용한다.

3.2 강건성 인식 selector

RA-QAOA는 후보 파라미터를 다음 구조로 점수화한다.

```
score =
  noisy_expected_cut / optimum
  - degradation_weight * |ideal_expected_cut - noisy_expected_cut| / optimum
  + success_weight * optimal_solution_probability
  - shot_variance_weight * shot_variance / optimum^2
  - sensitivity_weight * noise_sensitivity / optimum.
```

이 selector는 noisy expected cut과 optimal-solution probability를 보상하고, ideal-to-noisy degradation, shot variance, sensitivity across noise strengths를 penalty로 둔다. 이는 candidate parameter를 고르는 selector이지 새로운 quantum circuit이 아니다.

3.3 Baseline

비교 baseline은 ideal expected-cut selection, noisy expected-cut selection, random candidate selection, grid-center selection, continuous optimizer, multi-start optimizer이다. Optimizer baseline은 RA-QAOA를 raw-performance optimizer로 오해하지 않기 위해 중요하다.

3.4 시뮬레이션 및 proxy 실험

시뮬레이션 benchmark는 6, 8, 10 node random graph, QAOA depth p=1 및 p=2, depolarizing-style channel noise, readout error를 사용한다. 주요 요약은 같은 graph에서 robust minus baseline paired delta로 보고한다.

Hardware-like proxy는 특정 backend emulator가 아니다. QAOA gate load를 바탕으로 approximate one-qubit, two-qubit, readout error exposure를 final noise setting에 반영한다. 이 proxy는 덜 균일한 noise profile에서도 robustness signal이 유지되는지 확인하는 용도에 한정한다.

3.5 실제 QPU pilot

QPU pilot은 4-node square Max-Cut instance, p=2, circuit당 2048 shots를 사용한다. 각 Runtime job에는 `robust`, `noisy_expected`, `optimizer` 세 selector family를 제출한다.

항목	설정
backend	IBM `ibm_marrakesh`, IBM `ibm_kingston`
graph	4-node square Max-Cut
depth	p=2
shots	circuit당 2048
selectors	`robust`, `noisy_expected`, `optimizer`

항목	설정
repetitions	`ibm_marrakesh` : 3 jobs; `ibm_kingston` : 1 job
job ids	`d8a6rmop0eas73dpfntg` , `d8a6tq9789is73945510` , `d8a6tvdg7okc73epghh0` , `d8aprj2vnmpc73bqvnag`
metrics	QPU expected cut, optimal-solution success probability, within-job rank

Backend display order로 얻은 counts는 Max-Cut metric 계산 전에 graph-node order로 변환한다.

3.6 Metadata와 readout mitigation

저장된 QPU job에 대해 Runtime job properties를 조회하고 `job.inputs['pubs']`에서 transpiled circuit을 회수한다. Metadata table은 job-time calibration timestamp, recovered initial/final layout, active physical qubits, two-qubit pairs, active-qubit readout/gate error summary를 기록한다.

또한 independent per-qubit readout mitigation check를 적용한다. 각 selector circuit에서 final layout은 logical output bit를 physical readout error rate에 매핑한다. 이를 바탕으로 independent bit-flip confusion matrix를 만들고 pseudo-inverse correction을 사용한다. 이 check는 correlated readout, gate noise, crosstalk, coherent error, calibration drift를 모델링하지 않는다.

4. 결과

4.1 시뮬레이션 benchmark

60-graph p=2 benchmark에서 RA-QAOA는 noise sensitivity를 낮추고 success probability를 조금 높였지만 noisy expected cut은 낮췄다.

metric	robust - baseline	95% CI	해석
noisy E[C]	-0.136896	[-0.181157, -0.092636]	raw expected cut은 낮다
noise sensitivity	-0.393734	[-0.521034, -0.266434]	robust가 덜 noise-sensitive하다
success probability	+0.015303	[0.007746, 0.022859]	optimal-solution probability는 약간 높다

150-graph p=2 expanded benchmark에서도 sensitivity delta는 -0.412928, 95% CI [-0.494519, -0.331337]로 음수였다. 이는 selector가 단순히 우연히 좋은 case를 고른 것이 아니라 stability profile을 바꾸고 있음을 뒷받침한다.

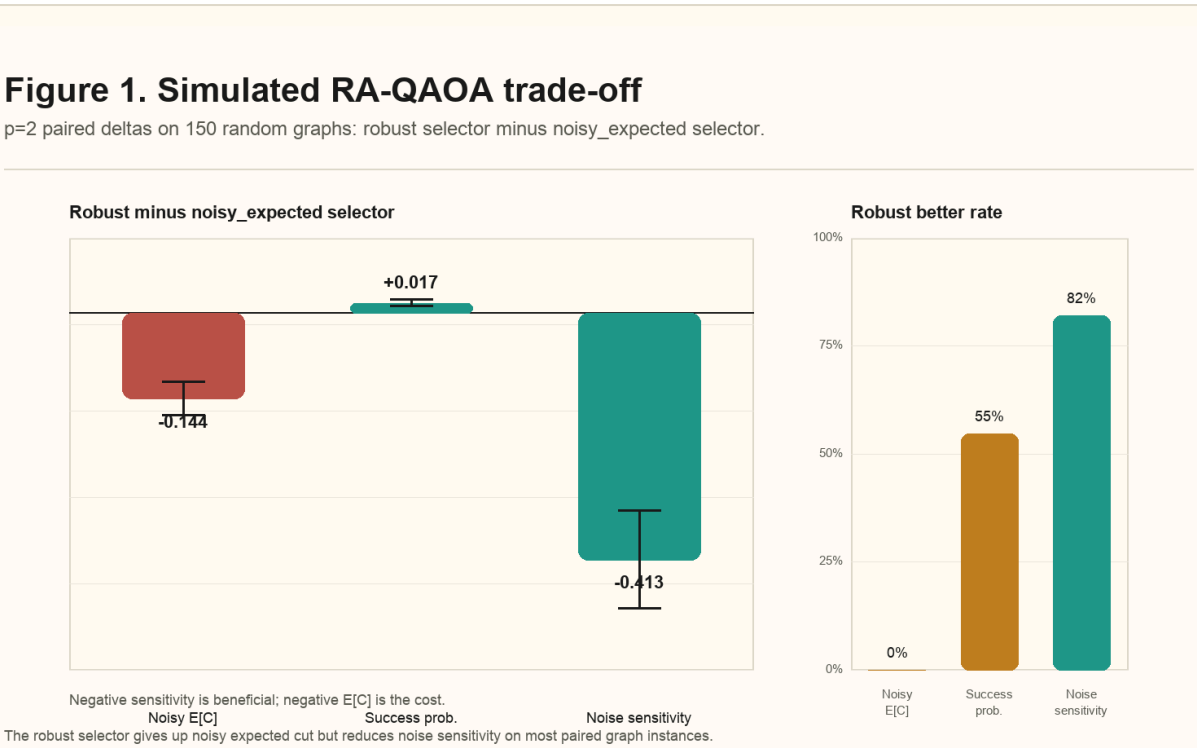

그림 1. p=2 시뮬레이션 benchmark trade-off. RA-QAOA는 noisy_expected 선택 대비 noise sensitivity를 낮추지만 noisy expected cut을 일부 포기한다.

4.2 Hardware-like proxy

Hardware-like proxy에서도 p=2의 sensitivity는 낮아졌다.

metric	value
p=2 sensitivity delta	-0.946489
95% CI	[-1.144908, -0.748070]
p=2 success delta	-0.008033
95% CI	[-0.019804, 0.003738]

이 proxy 결과는 stability signal을 뒷받침하지만 success-probability improvement claim을 뒷받침하지는 않는다.

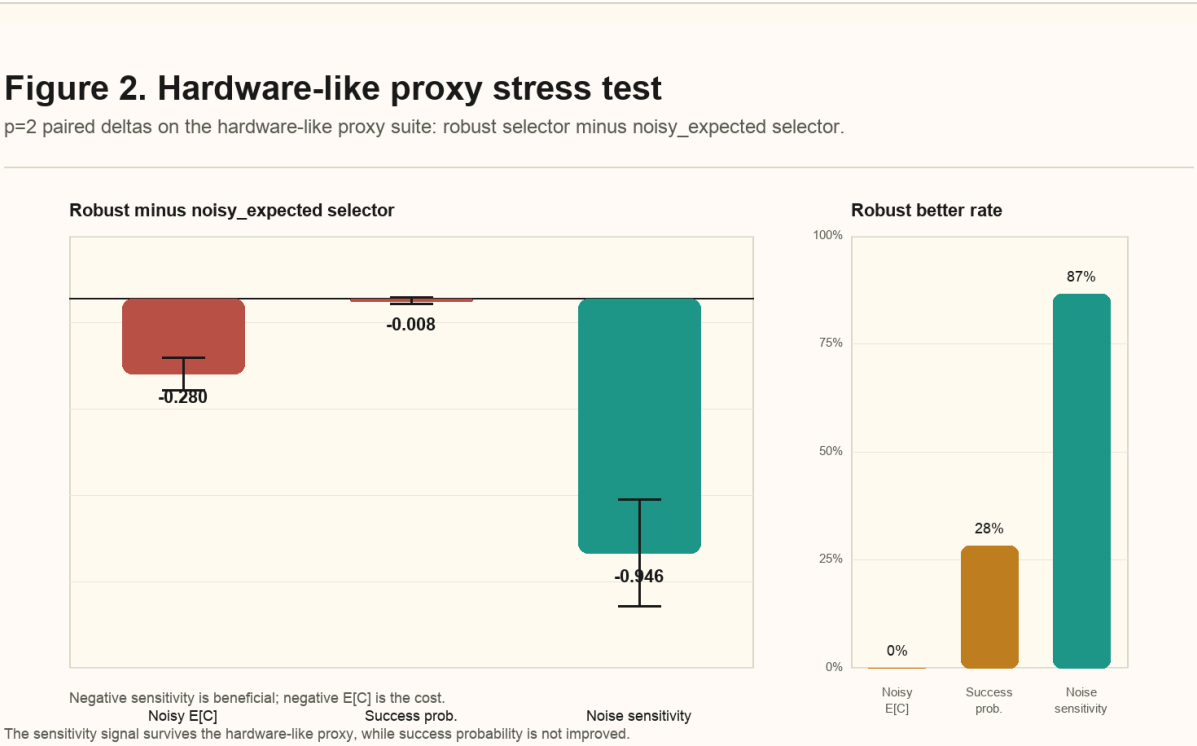

그림 2. Hardware-like proxy stress test. Proxy noise model에서도 robustness signal은 유지되지만 success probability 개선은 보이지 않는다.

4.3 Optimizer baseline

Optimizer baseline은 raw noisy E[C]와 success probability에서 RA-QAOA보다 높았다.

baseline	noisy E[C] delta	success delta	sensitivity delta
optimizer	-1.561254	-0.120209	-4.490405
multi-start optimizer	-1.561331	-0.120229	-4.490628

이 비교는 논문 framing의 핵심이다. RA-QAOA는 더 좋은 optimizer가 아니다. 안정성 중심 selector이며, 낮은 simulated sensitivity를 얻는 대신 raw performance를 일부 포기한다.

4.4 Real-QPU backend 비교

실제 QPU pilot에서는 `noisy_expected`가 테스트한 두 backend 모두에서 평균 winner였다.

backend	jobs	QPU E[C] winner	winner mean QPU E[C]	QPU success winner	winner mean QPU success
`ibm_kingston`	1	noisy_expected	3.583008	noisy_expected	0.805664
`ibm_marrakesh`	3	noisy_expected	3.666016	noisy_expected	0.847656

전체 4개 QPU job의 selector ranking은 다음과 같다.

selector	runs	backends	mean QPU E[C]	E[C] rank-1 count	mean QPU success	success rank-1 count
noisy_expected	4	2	3.645264	3	0.837158	3
optimizer	4	2	3.633301	0	0.828613	0
robust	4	2	3.597412	1	0.816284	1

Robust selector는 `ibm_marrakesh` 한 job에서 1위를 했지만, backend-level mean metric이나 overall mean에서는 1위가 아니다.

그림 3. 실제 QPU backend ranking. `noisy_expected`는 `ibm_kingston`과 `ibm_marrakesh` 모두에서 backend-level mean QPU expected cut 1위다.

4.5 Metadata와 layout

QPU jobs는 다음 backend/job metadata를 제공한다.

backend	jobs	job calibration timestamp	recovered layout source	active qubit examples
`ibm_marrakesh`	3	2026-05-26T00:06:48+09:00	`job.inputs['pubs']`	6;7;8;17 and 13;14;15;19
`ibm_kingston`	1	2026-05-26T21:24:44+09:00	`job.inputs['pubs']`	149;150;151;152

이 metadata는 pilot을 auditable하게 만든다. 다만 calibration이 selector ranking mismatch의 원인임을 직접 증명하지는 않는다.

4.6 Readout mitigation check

Independent per-qubit readout mitigation은 QPU metric을 높였지만 selector ranking을 바꾸지 않았다.

selector	runs	raw mean QPU E[C]	mitigated mean QPU E[C]	raw success	mitigated success	rank change
noisy_expected	4	3.645264	3.688433	0.837158	0.858515	1 -> 1
optimizer	4	3.633301	3.679195	0.828613	0.851114	2 -> 2
robust	4	3.597412	3.640847	0.816284	0.837750	3 -> 3

모든 within-job selector rank도 그대로 유지되었다. 따라서 관찰된 QPU ranking을 final independent readout error만으로 설명하기는 어렵다.

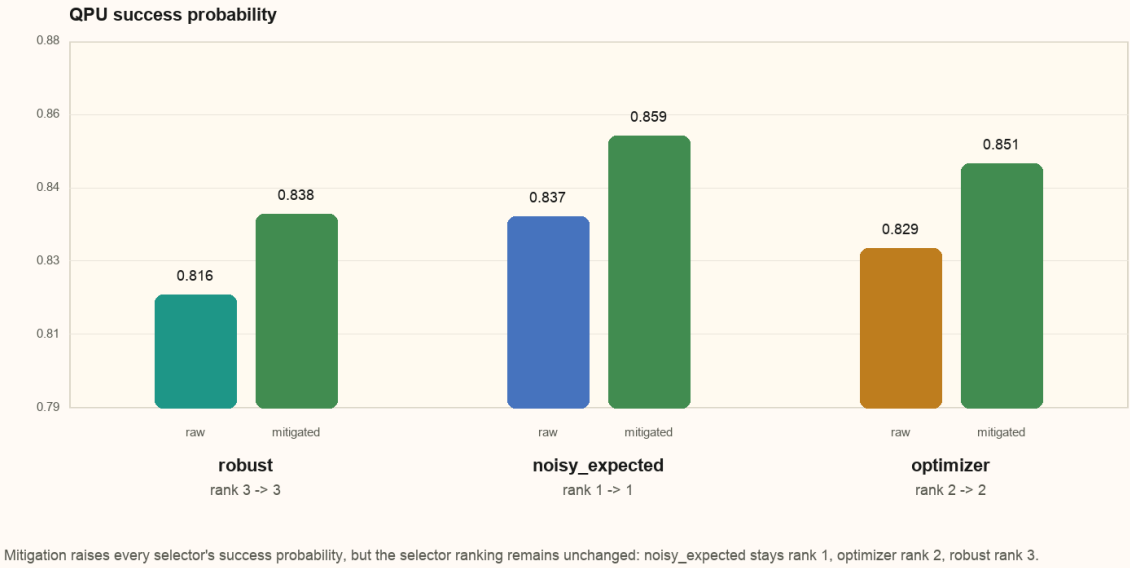

그림 4. Independent readout-mitigation check. 보정은 모든 selector의 success probability를 높이지만 overall selector ranking은 바꾸지 않는다.

4.7 Hardware-aware selector 후속 분석

QPU metadata는 실행위험을 명시적으로 반영하는 후속 selector도 제안한다. 이를 위해 저장된 backend-comparison table과 recovered layout/calibration table을 사용해 post-hoc hardware-aware selector 분석을 만들었다. 이 score는 simulator noisy expected cut과 simulator success probability를 약하게 보상하고, simulated noise sensitivity, transpiled depth, two-qubit gate count, active readout error, mean one-qubit gate error, mean two-qubit gate error, maximum two-qubit gate error를 penalty로 둔다. 각 feature는 같은 QPU job 안의 selector 사이에서 min-max 정규화한다.

저장된 4개 QPU job에서 hardware-aware score는 QPU expected-cut winner를 3/4 jobs에서 맞췄고, top-2 후보에는 4/4 jobs에서 실제 winner를 포함했다. 같은 row에서 simulator-only noisy expected-cut predictor는 top-1 winner를 0/4 jobs에서만 맞췄다. 그러나 단순 low-transpiled-depth baseline도 top-1 3/4를 달성했다. 따라서 이 후속 분석을 검증된 superior selector로 제시하면 안 된다. 더 객관적인 해석은 execution metadata가 selector objective에 들어가야 한다는 구체적 가설이며, 추가 QPU run 전에 weight를 사전 고정해야 한다는 것이다.

5. 논의

시뮬레이션 결과와 hardware 결과는 함께 해석해야 하지만, 하나의 승패 문장으로 합치면 안 된다. 시뮬레이션에서는 RA-QAOA가 설계 목적대로 동작한다. 즉 낮은 sensitivity 후보를 선택한다. 그러나 이는 측정 가능한 raw-performance cost를 동반한다. 실제 QPU에서는 같은 robust selector가 practical winner가 되지 않았다.

이 mismatch는 실험 실패가 아니라 본 논문의 주요 관찰이다. Selector가 selection 단계에서 사용한 simulated noise axis에 대해 robust하더라도, hardware가 추가하는 routing, calibration-dependent gate error, backend topology, finite-shot effect, unmodeled noise correlation 때문에 실제 장치에서 dominance로 이어지지 않을 수 있다.

Readout-mitigation check도 중요하다. 단순 per-qubit readout correction이 모든 selector rank를 그대로 유지했기 때문에, transfer gap이 순수한 final-measurement artifact일 가능성은 낮아진다. 남은 설명 후보는 gate error, routing difference, crosstalk, calibration drift, coherent error, simulator robustness objective와 hardware-relevant robustness 사이의 mismatch이다.

6. 한계

본 논문은 broad hardware claim을 의도적으로 피한다. 주요 한계는 다음과 같다.

1. QPU pilot은 4-node square graph와 p=2만 사용한다.
2. `ibm_marrakesh`는 3개 job이 있지만, `ibm_kingston`은 1개 job뿐이다.
3. 시뮬레이션 benchmark는 random graph family에 초점을 둔다. regular, weighted, planted, structured graph family는 별도 분석이 필요하다.
4. Hardware-like proxy는 full backend emulator가 아니며 routing, crosstalk, pulse schedule, queue-time drift를 모델링하지 않는다.
5. Readout mitigation model은 independent model이며 correlated measurement error를 포함하지 않는다.
6. RA-QAOA score weight는 설계 선택이다. 더 강한 claim 전에는 preregistered weight sweep이 필요하다.
7. Optimizer와 grid selector는 search budget이 다르므로 raw optimizer comparison은 algorithmic superiority를 통제해 증명하는 비교가 아니라 framing baseline으로 봐야 한다.
8. Hardware-aware 후속 분석은 4개 job에 대한 post-hoc 분석이다. 사전 고정한 low-depth baseline과 비교해 추가 검증해야 한다.

7. 결론

RA-QAOA는 QAOA parameter selection에서 stability-performance trade-off를 드러낸다. 시뮬레이션과 hardware-like proxy 실험에서 robust selector는 noise sensitivity를 낮춘다. 하지만 real QPU pilot에서는 noisy expected-cut selector가 테스트한 두 backend 모두에서 practical winner였다. Job-level metadata와 independent readout mitigation은 pilot을 더 auditable하게 만들며, 단순 final readout error가 ranking을 설명하지 못한다는 점을 보여준다.

가장 방어 가능한 결론은 simulated robustness와 real-device performance를 분리해서 보고해야 한다는 것이다. RA-QAOA가 유용한 이유는 현재 hardware performance를 지배해서가 아니라, simulator-to-hardware transfer gap을 측정 가능하게 만들기 때문이다.

재현성 자료

주요 자료:

- `reports/robust-qaoa-expanded-benchmark.md`

- `reports/robust-qaoa-hardware-like-noise.md`
- `reports/robust-qaoa-optimizer-baseline.md`
- `reports/robust-qaoa-multistart-optimizer.md`
- `reports/qpu_runs/ra_qaoa_qpu_backend_comparison.md`
- `reports/qpu_runs/ra_qaoa_qpu_metadata.md`
- `reports/qpu_runs/ra_qaoa_qpu_readout_mitigation.md`
- `reports/qpu_runs/hardware_aware_qaoa_selector.md`

재생성 명령:

```
python examples/ra_qaoa_qpu_backend_compare.py
python examples/ra_qaoa_qpu_paper_table.py
python examples/ra_qaoa_qpu_metadata_table.py
python examples/ra_qaoa_qpu_readout_mitigation.py
python examples/hardware_aware_qaoa_selector.py
```

참고문헌

`paper/references.bib`를 참조한다.