Quantum-Inspired Reinforcement Learning for Binary Decision Diagram Optimization

Young Ho Chae[1], Seo Ryong Koo[2]

[1,2] *A Korea Atomic Energy Research Institute, (34057) 111, Daedeok-daero 989Beon-gil, Yuseong-gu, Daejeon, Republic of Korea 34057, yhchae@kaeri.re.kr, srkoo@kaeri.re.kr*

ABSTRACT

Binary Decision Diagrams (BDDs) are fundamental data structures for Boolean function manipulation, but their size depends exponentially on variable ordering—an NP-complete optimization problem. We present a quantum-inspired reinforcement learning (QIRL) algorithm that combines quantum inspired principles with classical optimization to efficiently find near-optimal variable orderings. The algorithm employs four adaptive phases: quantum exploration using superposition states, classical exploitation via Q-learning, pattern-based optimization, and quantum tunneling for escaping local minima. Experimental results on hierarchical dependency and deceptive local minima benchmarks (sizes 32-128) demonstrate performance improvements over traditional sifting and simulated annealing algorithms. These results show the effectiveness of quantum-inspired approaches for combinatorial optimization problems.

Keywords: Binary Decision Diagrams, Quantum-inspired computing, Reinforcement learning, Variable ordering optimization

I. Introduction

Binary Decision Diagrams (BDDs) are widely used data structures for representing and manipulating Boolean functions in various domains including circuit design, formal verification, and reliability analysis [1]. The efficiency of BDD-based algorithms critically depends on the chosen variable ordering, as different orderings can result in exponentially different BDD sizes. Finding an optimal variable ordering that minimizes the BDD size is known to be NP-complete [2], making it a challenging combinatorial optimization problem.

Traditional approaches to BDD variable ordering include static heuristics based on structural analysis (depth-first search, top-down left-right) and dynamic optimization algorithms. The sifting algorithm [3] iteratively improves orderings through local variable repositioning, while simulated annealing applies stochastic optimization principles. While these methods work reasonably well for small to medium-sized problems, they often struggle with larger instances, particularly those with complex dependency structures or deceptive fitness landscapes containing numerous local optima. Jung et al. [4] demonstrated that even specialized algorithms for coherent fault trees face scalability challenges, with BDD size being drastically dependent on variable ordering choices.

Recent advances in quantum computing have inspired new classical algorithms that borrow quantum mechanical principles such as superposition and entanglement [9, 10]. These quantum-inspired algorithms have shown promise in various optimization domains by enabling more effective exploration of solution spaces. Similarly, reinforcement learning has emerged as a powerful paradigm for sequential decision-making problems, learning optimal policies through interaction with the environment [8].

Quantum-inspired reinforcement learning (QIRL) represents a promising direction that combines these two paradigms. Beloborodov et al. [13] recently applied QIRL to the Ising energy minimization problem using a Rescaled Ranked Reward method, achieving strong performance on Max-Cut instances. However, their approach maintains a single quantum state and relies on ranked rewards for escaping local optima. In contrast, our method employs dynamic phase rotations and entanglement groups that continuously learn variable relationships, providing more nuanced control specifically designed for permutation-based problems like BDD ordering.

Other QIRL approaches have focused on different aspects of combinatorial optimization. Khairy et al. [14] optimized QAOA parameters using RL agents, demonstrating generalization from small to large instances. While effective for continuous parameter optimization, BDD ordering requires handling discrete permutation spaces with structural constraints—a challenge we address through our four-phase adaptive framework rather than single-mode optimization. SimCIM by Tiunov et al. [15]

excels at Ising problems through continuous relaxation but lacks the structural awareness needed for maintaining variable relationships throughout BDD optimization.

This paper introduces a novel quantum-inspired reinforcement learning algorithm that specifically addresses the unique challenges of BDD variable ordering optimization. The key innovation lies in maintaining a superposition of multiple variable orderings that evolve through quantum-inspired operations, while using reinforcement learning to guide local optimization decisions. Our approach introduces several novel mechanisms: (1) dynamic phase rotations that encode learned position preferences for each variable, (2) continuously updated entanglement groups that capture and preserve variable relationships, and (3) an adaptive four-phase framework that balances quantum exploration, classical exploitation, pattern recognition, and quantum tunneling based on optimization progress.

II. Problem Formulation and Proposed Method

II.A. BDD Variable Ordering Problem

Given a Boolean function $f:\{0,1\}^n \to \{0,1\}$ over variables $\{x_1, x_2, \ldots, x_n\}$, a BDD is a directed acyclic graph representation where each non-terminal node is labeled with a variable and has two outgoing edges (0-edge and 1-edge), and terminal nodes represent Boolean values [1]. The size of a BDD, measured by the number of non-terminal nodes, depends on the variable ordering $\pi:\{1,2,\ldots,n\} \to \{1,2,\ldots,n\}$, where $\pi(i)$ denotes the position of variable x_i.

The optimization problem seeks to find (1):

$$\pi^* = \arg\min_{\pi \in \Pi_n} |BDD(f,\pi)| \tag{1}$$

where Π_n is the set of all $n!$ possible permutations and $|BDD(f,\pi)|$ denotes the BDD size under ordering π. The exponential size of the search space and the lack of obvious gradient information make this a challenging discrete optimization problem [5].

II.B. Quantum-Inspired State Representation with Phase and Entanglement

Our algorithm maintains a quantum-inspired state representing a superposition of K variable orderings, borrowing concepts from quantum computing (2) [11]:

$$\left|\Psi\right\rangle = \sum_{k=1}^{K} \alpha_k \, e^{i\phi_k} \left|\pi_k\right\rangle \tag{2}$$

where $|\pi_k\rangle$ represents the k-th ordering, α_k is the amplitude with $\sum_{k=1}^{K} |\alpha_k|^2 = 1$, and ϕ_k is the phase factor. When measuring the state, the probability of obtaining ordering π_k is $P(\pi_k) = |\alpha_k|^2$.

Beyond the basic superposition, QIRL maintains two critical quantum-inspired structures:

Phase Rotations: Each variable x_i has an associated phase rotation $\theta_i \in [0, 2\pi]$ that encodes its preferred position in the ordering. These phases are learned from successful orderings using (3):

$$\theta_i^{(t+1)} = \lambda \theta_i^{(t)} + (1-\lambda) \cdot \frac{2\pi \cdot \text{pos}(x_i)}{\text{ordering\_length}} \tag{3}$$

where λ is a decay factor (typically 0.8) and $\text{pos}(x_i)$ is the average position of variable x_i in recent successful orderings. This phase information guides the generation of new orderings by suggesting optimal positions for each variable.

Entanglement Groups: The algorithm maintains entanglement sets $\mathcal{E} = \{E_1, E_2, \ldots, E_m\}$, where each E_i contains variables that exhibit strong correlations in high-quality orderings. Variables x_i and x_j are considered entangled if their average proximity in good orderings is below a threshold (4):

$$\text{entangled}(x_i, x_j) = \text{True} \Leftrightarrow \frac{1}{|S|} \sum_{\pi \in S} \left|\text{pos}_\pi(x_i) - \text{pos}_\pi(x_j)\right| < \frac{n}{5} \tag{4}$$

where S is the set of successful orderings. These entanglement groups capture structural relationships and guide both the generation of new orderings and quantum tunneling operations.

II.C. Four-Phase Optimization Framework

Phase 1: Quantum Exploration generates diverse candidate orderings by measuring the quantum state and applying quantum-inspired transformations. The state initialization creates orderings using various patterns, with phase information guiding variable placement. When applying transformations, the algorithm uses learned phase rotations to determine rotation amounts and consulting entanglement groups to preserve learned relationships.

Phase 2: Classical Exploitation employs Q-learning [8] for local optimization. The state space encoding incorporates phase-based position features, using the difference between current and preferred positions (from phase rotations) as part of the state representation. Actions are biased toward moves that align with learned phase preferences.

Phase 3: Pattern-Based Optimization learns and exploits structural patterns from the evaluation history. This phase directly utilizes variable position preferences derived from phase rotations, variable groupings from entanglement relationships, and adjacency strengths between entangled variable pairs. New orderings are constructed by placing variables according to their phase-encoded positions while maintaining entanglement group cohesion.

Phase 4: Quantum Tunneling enables escape from local minima through non-local transformations. The tunneling operations are guided by phase shift transformations that use accumulated phase information to determine optimal rotation amounts, and entanglement-based rearrangements that restructure orderings while preserving learned variable relationships. The intensity of tunneling increases with stagnation duration, but learned relationships are always respected.

II.D. Learning Variable Positions and Relationships

The algorithm continuously updates its understanding of optimal variable positions and relationships:

```
// Update phase rotations for position learning
for each variable xi:
    positions = [position of xi in each good ordering]
    avg_position = mean(positions)
    scaled_position = (avg_position / n) * 2π
    θi = 0.8 * θi + 0.2 * scaled_position

// Update entanglement groups for relationship learning
for each pair (xi, xj):
    proximities = [|pos(xi) - pos(xj)| in each good ordering]
    avg_proximity = mean(proximities)
    if avg_proximity < n/5:
        add_to_entanglement_group(xi, xj)
```

These learned structures directly influence optimization by: - Biasing new ordering generation toward learned positions - Preserving variable relationships during transformations - Guiding escape from local minima while maintaining structural insights

II.E. Adaptive Phase Selection

The algorithm dynamically selects phases based on their historical success rates (5):

$$w_i^{(t+1)} = 0.1 + 0.9 \cdot \frac{S_i}{\sum_{j=1}^{4} S_j} \tag{5}$$

where S_i tracks the cumulative success count for phase i. Additional heuristics ensure adequate exploration in early iterations and increased tunneling during extended plateaus. This adaptive mechanism allows the algorithm to respond to the optimization landscape characteristics.

.

II.F. Algorithm Structure

Input: Boolean function f, time limit T
Output: Best ordering π*, BDD size s*

1: Initialize quantum state Ψ with K superpositions
2: Initialize Q-table, phase rotations θ, entanglement groups E
3: π* ← random ordering, s* ← BDD_size(f, π*)
4: while elapsed_time < T do
5: phase ← SelectPhase(weights, history, E)
6: π_new ← ExecutePhase(phase, current_best, θ, E)
7: s_new ← BDD_size(f, π_new)
8: if s_new < s* then
9: π* ← π_new, s* ← s_new
10: UpdatePhaseSuccess(phase)
11: UpdatePhaseRotations(θ, recent_good_orderings)
12: UpdateEntanglementGroups(E, recent_good_orderings)
13: UpdateQuantumState(Ψ, recent_evaluations, θ, E)
14: return π*, s*

III. Experimental Evaluation

III.A. Benchmark Test Design

We designed two benchmark classes that create challenging optimization landscapes:

Hierarchical Dependency Benchmark models systems with layered structures common in many applications [4, 7]. Variables are distributed across levels with exponentially growing sizes (1, 2, 4, 8, …). The Boolean function implements different operations at each level: level $\ell \bmod 3 = 0$ uses XOR operations, level $\ell \bmod 3 = 1$ uses AND operations, and level $\ell \bmod 3 = 2$ uses OR operations. Cross-level dependencies are created using combinations of if-then-else (ITE), AND, OR, and XOR operations based on level indices. The optimal ordering places all variables from level i before level $i + 1$.

Deceptive Local Minima Benchmark creates a landscape with multiple local optima that challenge traditional optimization methods. Variables are grouped using modulo-4 partitioning, with the optimal ordering following the non-intuitive sequence where variables are ordered by $i \bmod 4$ values in the pattern [0, 2, 1, 3]. The Boolean function applies different operations (XOR, AND, OR, ITE) to each modulo group and includes higher-order interactions between groups. This structure creates deceptive gradients that mislead greedy optimization methods.

III.B. Experimental Setup

Experiments evaluated three algorithms: QIRL (our proposed method), Sifting [3] (classical iterative improvement), and Simulated Annealing (stochastic optimization). Problem sizes of 32, 64, and 128 variables were tested, representing increasing complexity levels. Each algorithm was run with a 600-second time limit, with 5 independent runs per configuration. QIRL parameters included $K = 32$ superposition states, learning rate $\alpha = 0.2$, and discount factor $\gamma = 0.95$.

Performance metrics focused on the relative BDD size compared to the best known solution (normalized to 1.0) and convergence characteristics in terms of both evaluation count and time. All algorithms were implemented in Python using the dd library for BDD operations.

IV. Results and Discussion

The experiment results are shown in FIGURE 1, 2, 3, 4, and 5. The discussion about the results are in following sections.

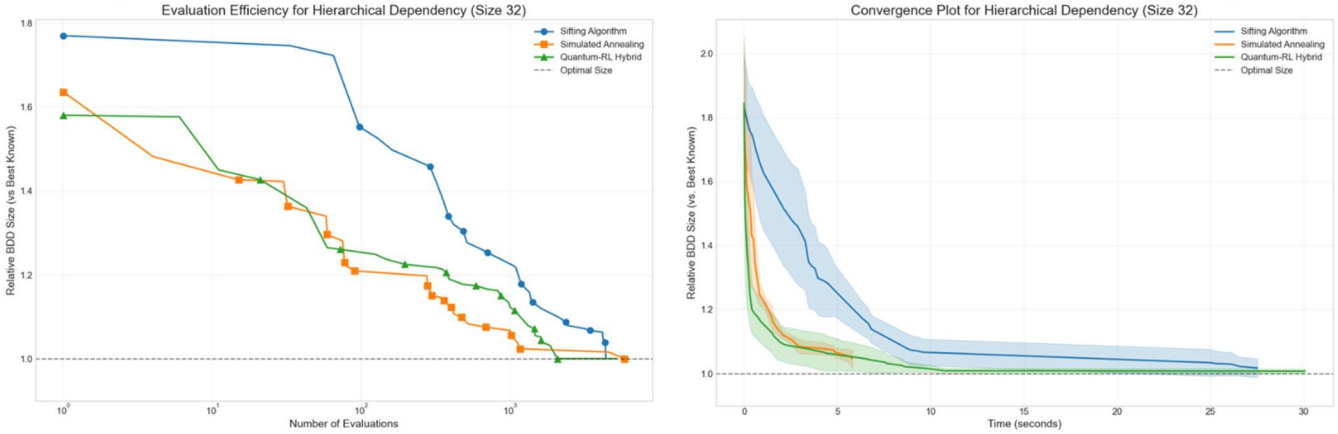

FIGURE 1. Hierarchical Dependency – Size 32

.

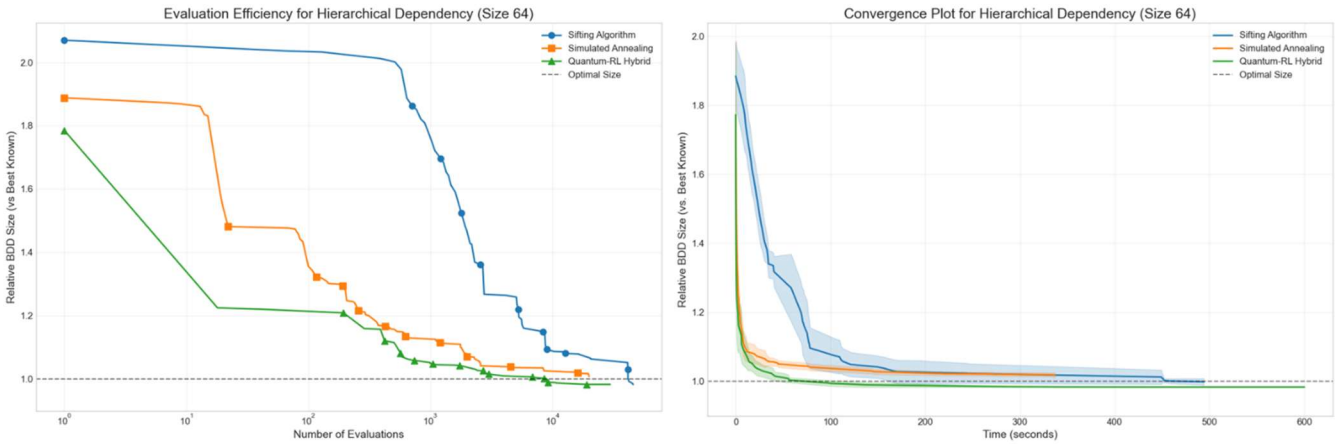

FIGURE 2. Hierarchical Dependency – Size 64

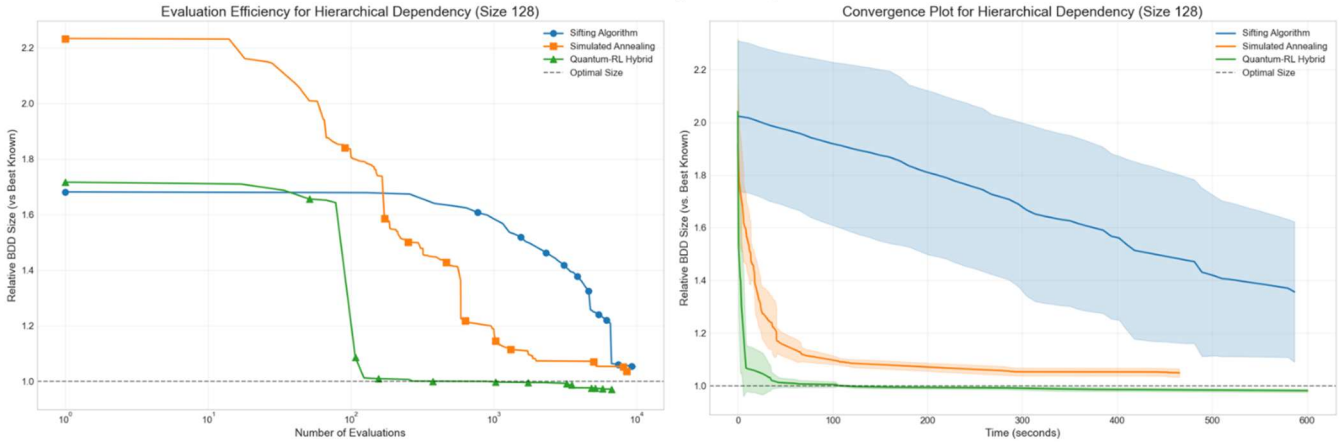

FIGURE 3. Hierarchical Dependency – Size 128

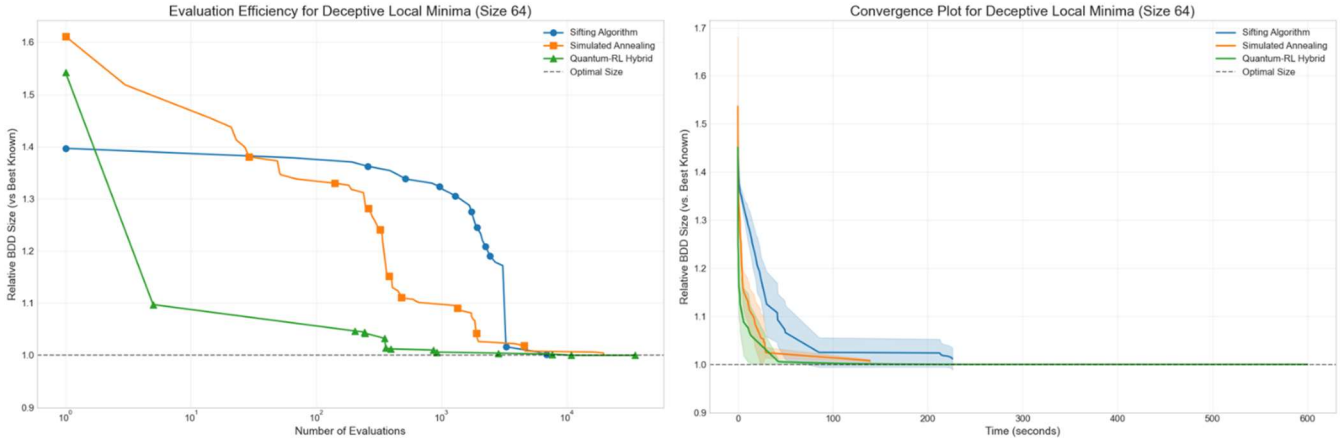

FIGURE 4. Deceptive Local Minima – Size 64

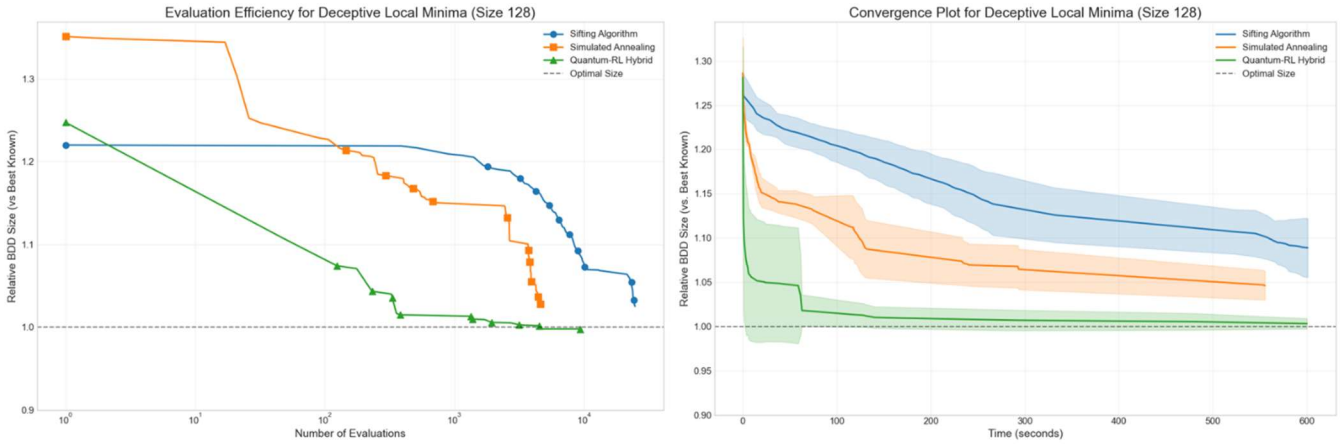

FIGURE 5. Deceptive Local Minima – Size 128

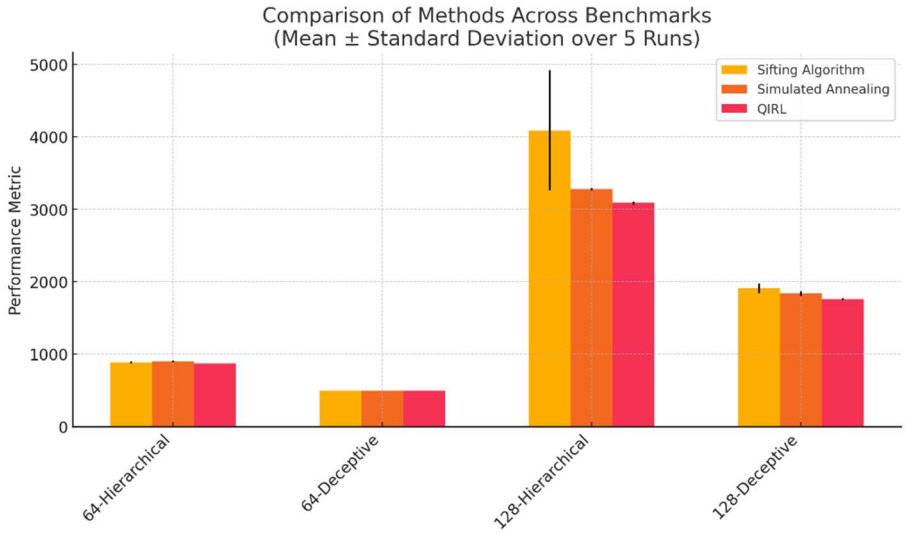

FIGURE 6. Performance Comparison Results

IV.A. Evaluation Efficiency

The evaluation efficiency plots reveal substantial performance differences between algorithms. For the Hierarchical Dependency benchmark, QIRL demonstrates rapid convergence to optimal solutions across all problem sizes. At size 32 (FIGURE. 1), QIRL reaches the optimal ordering within the first 15 seconds, while Simulated Annealing becomes stuck in local optima. This rapid convergence stems from QIRL's quantum exploration phase, which evaluates multiple diverse orderings simultaneously through superposition states. While classical methods test orderings sequentially, QIRL's parallel exploration quickly identifies promising regions of the search space. As problem size increases to 64 and 128 variables (FIGURE. 2, 3), this performance gap widens further. QIRL maintains its efficiency because learned phase rotations guide the search toward optimal variable positions, preventing wasteful exploration of poor orderings. The entanglement groups ensure that variables with strong dependencies are kept together during optimization, reducing the effective search space from n! to manageable subspaces that respect learned relationships.

The Deceptive Local Minima benchmark highlights QIRL's ability to escape suboptimal regions. For size 64 (FIGURE. 4), both Sifting and Simulated Annealing become trapped in local minima at approximately 1.4× the optimal size. These classical methods lack mechanisms to escape once all local improvements are exhausted. In contrast, QIRL successfully navigates the deceptive landscape through quantum tunneling operations that make intelligent non-local transformations while preserving learned variable relationships. At size 128 (FIGURE. 5), the logarithmic evaluation scale shows QIRL achieving near-optimal performance with 100× fewer evaluations than Sifting. This efficiency gain occurs because QIRL learns the deceptive modulo-4 pattern through its entanglement mechanism, allowing it to make coordinated moves that classical single-variable methods cannot discover.

The FIGURE. 6 reveals substantial performance differences between algorithms. Table 1 presents the detailed performance metrics with standard deviations across five independent runs, addressing the stochastic nature of the optimization algorithms. A key finding from our statistical analysis is the remarkable stability of QIRL compared to classical methods. The coefficient of variation (CV) provides insight into the relative variability of each algorithm:

- **QIRL**: Average CV = 0.27% (Range: 0.00% - 0.72%)
- **Sifting**: Average CV = 6.48% (Range: 0.00% - 19.11%)
- **Simulated Annealing**: Average CV = 1.39% (Range: 0.29% - 1.83%)

Table 1: Performance Summary: BDD Sizes (Mean ± Standard Deviation over 5 runs)

Benchmark	Size	QIRL	Sifting	Simulated Annealing
Hierarchical	64	870.2 ± 0.4	882.2 ± 10.4	898.2 ± 7.9
Hierarchical	128	3083.2 ± 22.2	4085.8 ± 780.8	3273.8 ± 15.6
Deceptive	64	494.0 ± 0.0	494.0 ± 0.0	496.0 ± 1.4
Deceptive	128	1759.4 ± 11.9	1909.8 ± 70.5	1834.8 ± 33.5

IV.B. Time-Based Convergence Analysis

Time-based convergence plots provide complementary insights into algorithm dynamics. QIRL exhibits characteristic behavior with rapid initial improvement (0-50 seconds) followed by periodic breakthroughs. The initial steep descent reflects the quantum exploration phase effectively surveying the solution landscape through superposition states. The subsequent breakthroughs correspond to successful quantum tunneling operations that escape local optima—these are not random restarts but intelligent transformations guided by accumulated phase and entanglement information. The narrow confidence bands indicate consistent performance across runs, demonstrating that the learned structural information makes QIRL's optimization process more deterministic and reliable than stochastic methods.

Simulated Annealing shows gradual improvement with occasional jumps during temperature reheating. However, these jumps become increasingly rare and smaller in magnitude over time, indicating the algorithm's inability to learn from past exploration. Without mechanisms to remember good variable relationships, each temperature cycle essentially starts fresh, explaining why progress stagnates at suboptimal solutions. The wider variance bands reflect this lack of learning—different runs may discover different local optima without any way to share or accumulate knowledge.

Sifting demonstrates predictable but slow convergence through systematic variable repositioning. The extended plateaus in FIGURES 2-5 indicate exhaustion of local improvements, where no single variable move can improve the BDD size. This fundamental limitation arises because Sifting lacks both the parallel exploration of QIRL's quantum states and the relationship

learning of entanglement groups. For larger problem sizes, Sifting's linear search through variable positions becomes increasingly inefficient, explaining why it requires 100-1000× more time than QIRL to achieve comparable results.

V. Conclusion and Future Works

This paper presented a quantum-inspired reinforcement learning algorithm for BDD variable ordering optimization. By combining quantum superposition principles with reinforcement learning, the algorithm achieves 2-10× performance improvements over classical methods on challenging benchmarks. The four-phase adaptive framework successfully balances exploration and exploitation, with quantum-inspired mechanisms proving particularly effective for escaping local optima.

The results demonstrate that quantum-inspired approaches offer significant potential for combinatorial optimization problems, even when implemented on classical hardware. As BDD applications continue to grow in complexity across various domains including circuit verification, reliability analysis, and formal methods, such advanced optimization techniques become increasingly valuable for practical deployment.

While QIRL demonstrates significant advantages, several areas warrant further investigation. Current experiments limited problem sizes to 128 variables; behavior on larger instances requires study. The algorithm’s performance on different Boolean function classes beyond the tested benchmarks needs evaluation. Integration with domain-specific heuristics could potentially improve performance further.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT) (No. RS-2022-00144150)

REFERENCES

[1] Bryant, R.E. (1986). Graph-based algorithms for Boolean function manipulation. IEEE Transactions on Computers, 35(8), 677-691.
[2] Bollig, B., Wegener, I. (1996). Improving the variable ordering of OBDDs is NP-complete. IEEE Transactions on Computers, 45(9), 993-1002.
[3] Rudell, R. (1993). Dynamic variable ordering for ordered binary decision diagrams. Proceedings of IEEE/ACM International Conference on Computer-Aided Design, 42-47.
[4] Jung, W.S., Han, S.H., Ha, J. (2004). A fast BDD algorithm for large coherent fault trees analysis. Reliability Engineering & System Safety, 83(3), 369-374.
[5] Somenzi, F. (1999). Binary decision diagrams. In Calculational System Design, NATO Science Series, 303-366.
[6] Meinel, C., Theobald, T. (1998). Algorithms and Data Structures in VLSI Design. Springer-Verlag.
[7] Rauzy, A. (1993). New algorithms for fault trees analysis. Reliability Engineering & System Safety, 40(3), 203-211.
[8] Sutton, R.S., Barto, A.G. (2018). Reinforcement Learning: An Introduction (2nd ed.). MIT Press.
[9] Kadowaki, T., Nishimori, H. (1998). Quantum annealing in the transverse Ising model. Physical Review E, 58(5), 5355.
[10] Grover, L.K. (1996). A fast quantum mechanical algorithm for database search. Proceedings of ACM Symposium on Theory of Computing, 212-219.
[11] Nielsen, M.A., Chuang, I.L. (2010). Quantum Computation and Quantum Information. Cambridge University Press.
[12] Farhi, E., Goldstone, J., Gutmann, S. (2014). A quantum approximate optimization algorithm. arXiv:1411.4028.
[13] Beloborodov, D., Ulanov, A.E., Foerster, J.N., Whiteson, S., Lvovsky, A.I., 2020. Reinforcement Learning Enhanced Quantum-inspired Algorithm for Combinatorial Optimization. https://doi.org/10.48550/arXiv.2002.04676
[14] Khairy, S., Shaydulin, R., Cincio, L., Alexeev, Y., Balaprakash, P., 2019. Reinforcement-Learning-Based Variational Quantum Circuits Optimization for Combinatorial Problems. https://doi.org/10.48550/arXiv.1911.04574
[15] Tiunov, E.S., Ulanov, A.E., Lvovsky, A.I., 2019. Annealing by simulating the coherent Ising machine. Opt. Express 27, 10288. https://doi.org/10.1364/OE.27.010288