Hierarchical Graph Representation for Multi-Chain Blockchain Routing

Tirth Joshi
Katz School of Health and Science
Yeshiva University
New York, USA
tjoshi1@mail.yu.edu

Honggang Wang
Department of Graduate Computer Science and Engineering
Yeshiva University
New York, USA
Honggang.wang@yu.edu

***Abstract*—Existing blockchain graph models treat assets and contracts as flat networks, thereby missing hierarchical and logical dependencies among tokenized entities. We propose a hierarchical floor-and-building representation for multi-chain ecosystems, modeling inter-asset dependencies as a bounded-depth AND/OR graph. The associated pathfinding is formulated as an NP-hard optimization problem; in the general case, it is tackled using a hybrid approach that combines Dijkstra-style greedy search with recursive dependency reconstruction. This structure enables scalable routing and reasoning across chains with improved locality and tractability. Experiments on large multi-chain datasets show sub-linear growth in runtime and substantial reduction in search space compared with flat-graph Dijkstra baselines. Experiments on 63,412 tokens, 191,085 edges, and 11,726 bridges, show an 8x runtime reduction and 5% deviation from optimal solutions.**

***Index Terms*—Decentralized Finance (DeFi), Cross-chain Routing, Graph Theory, AND/OR Graphs, Multi-layer Networks, Pathfinding, Liquidity Networks, Blockchain Analytics.**

I. Introduction

Blockchain ecosystems have evolved into large, heterogeneous networks of tokens, contracts, and bridges that resemble multi-layer communication systems rather than isolated ledgers. Assets no longer exist as independent units but as compositional derivatives,liquidity-pool (LP) shares, vault receipts, and synthetic tokens, stacked through recursive dependency structures [1], [2]. As liquidity and state information flow across chains, these interactions form deeply entangled topologies whose complexity challenges conventional graph-based analysis and routing [3]–[5].

Existing models cannot simultaneously represent hierarchical dependencies and optimize routing across interconnected chains. Conventional flat-graph approaches used in cross-chain routing and swap analysis [6], [7] treat all entities as peers, ignoring that many Decentralized Finance (DeFi) instruments depend on lower-level assets. This omission leads to redundant traversal, excessive edge density, and incorrect path construction when multiple prerequisites must be satisfied conjunctively. Moreover, prior constrained-path formulations [20], [21] model dependencies as side constraints rather than as part of the search structure itself, preventing scalable reasoning across compositional systems.

We address these issues by introducing a **hierarchical floor-and-building representation** for multi-chain blockchain networks. Each blockchain is modeled as a building whose internal floors correspond to increasing abstraction levels, from base tokens and liquidity pools to higher-order derivatives, while bridges act as inter-building connectors characterized by latency, cost, and reliability parameters. This architecture enables localized traversal within each floor before cross-chain transitions, improving both storage locality and routing tractability.

Routing in this framework is formalized as search over a weighted **AND/OR graph**, where nodes represent conjunctive dependencies (AND) or alternative paths (OR). The general minimum-cost path problem on such graphs is NP-hard [18], [19]. The NP-hardness of dependency-constrained routing follows from a reduction of 3-SAT to AND/OR path minimization, where literals form OR-nodes and clause satisfaction corresponds to AND aggregation. By constraining the conjunctive depth to $d_{\text{AND}} \leq 2$, practical blockchain networks become polynomially solvable, enabling efficient dependency-aware routing.

Our contributions are summarized as follows:

- A hierarchical representation for multi-chain networks enabling constant-time floor-level access and semantic locality.

979-8-3315-7111-5/26/$31.00 ©2026 IEEE

- A bounded-AND pathfinding formulation ensuring tractability in dependency-rich routing.
- A greedy hybrid algorithm combining Dijkstra's efficiency with recursive dependency reconstruction, extending beyond constrained shortest-path models [20], [21].

II. Related Work and Literature Review

Research in DeFi increasingly examines how compositional assets and cross-chain systems interact. Existing studies, however, treat structure and routing separately, lacking a unified hierarchical abstraction.

A. Token Composition and Structural Graphs

Harrigan et al. [1] and Kitzler et al. [2] introduced *token composition graphs* built from EVM logs, showing that DeFi assets form deeply nested dependency chains. Luo et al. [8], [9] extended this to graph-representation learning for protocol clustering, while Chiu et al. [10] modeled macro token flows using input–output networks. Although structurally insightful, these approaches rely on flat storage and lack scalable, query-efficient routing.

B. Cross-Chain Routing and Interoperability

Cross-chain computation has been explored through surveys and optimization studies. Ou et al. [3] summarized relay and HTLC-based mechanisms; Zhang et al. [4] and Lin et al. [5] modeled bridge security via heterogeneous graphs; Huang et al. [7] proposed resilient-overlay routing, and Muhammad & Kristensen [6] formulated bridge selection as an ILP. These frameworks improve path accuracy but incur high computational cost and ignore compositional semantics.

C. Hierarchical and Logical Graph Modeling

Hierarchical routing in transportation and spatial networks [11], [12] and multi-layer graph theory [13], [14] inspire our floor-based segmentation of chains. Meanwhile, AND/OR graph reasoning [15]–[18] proves that bounded conjunctive depth yields polynomial-time solvability, a property we exploit for tractable DeFi pathfinding. Recent DeFi graph surveys [22], [23] address scale via indexing but omit logical dependency handling. Unlike constrained shortest path formulations [20], [21], our method embeds logical dependencies directly into the search space rather than treating them as side constraints.

TABLE I
Representative Literature and Scope Comparison

Work	Focus	Limitation
Harrigan et al. (2024)	Token graphs	Flat only
Kitzler et al. (2021)	Protocol nesting	No routing
Muhammad & Kristensen (2023)	Cross-chain ILP	Expensive
Huang et al. (2024)	Overlay routing	No semantics
Lin et al. (2025)	Bridge GNNs	Security only
This work	Hierarchical routing	Unified, scalable

D. Positioning

Unlike prior efforts that treat either composition or routing, our framework integrates both through a hierarchical, bounded-AND design that achieves structural interpretability, constant-time retrieval, and polynomial routing complexity.

III. Model and Representation

A. Hierarchical Abstraction

We formalize the multi-chain blockchain ecosystem as a hierarchical structure where each chain operates as an independent domain with internal dependency layers. Let the entire system be represented as a directed multigraph

$$G = (C, V, E, F, \Phi, \Psi),$$

where:

- $C = \{c_1, c_2, \ldots, c_n\}$ is the set of blockchains (*buildings*);
- $V = \bigcup_i V_i$ represents all contract or token nodes across chains;
- E is the set of directed edges encoding functional relations (swaps, bridges, liquidity connections);
- $F = \{f_1, f_2, \ldots, f_m\}$ denotes abstraction levels (*floors*) within each chain;
- $\Phi : V_i \to F$ maps each node to its abstraction level;
- $\Psi : E \to C_i \times C_j$ labels each edge as intra-chain or cross-chain.

Each *building* c_i contains multiple *floors*, where lower floors host primitive tokens and upper floors represent higher-order derivatives, liquidity shares, or bundled assets. Edges exist both within and across floors, forming

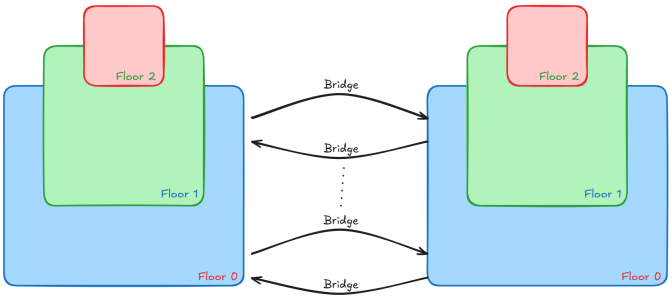

Fig. 1. Structure of the hierarchical multi-chain DeFi framework.

a structured topology that preserves semantic hierarchy. A cross-chain bridge is modeled as an edge (u, v) where $\Psi(u, v) = (c_i, c_j)$ and $\Phi(u) = \Phi(v)$, ensuring consistent abstraction across domains.

B. Adjacency and Memory Layout

To support scalable traversal, each building maintains a layered adjacency block:

$$\mathcal{A}_{c_i} = [A_{f_1}, A_{f_2}, \ldots, A_{f_m}],$$

where A_{f_k} stores intra-floor connectivity in compressed sparse-row (CSR) form. This arrangement allows $O(1)$ retrieval for nodes within a floor and $O(\log n)$ cross-floor updates, improving cache locality and traversal speed in large-scale graphs.

C. Hierarchical Semantics

Each edge (u, v) is associated with a tuple

$$w(u, v) = (\text{cost},\ \text{time},\ \text{reliability}),$$

where *cost* denotes transactional expense, *time* denotes confirmation latency, and *reliability* captures bridge or protocol stability. Multi-dimensional weights are normalized to a scalar via a user-defined utility function

$$\mu(u, v) = \alpha_c\, w_c + \alpha_t\, w_t + \alpha_r\, (1 - w_r),$$

where $\alpha_c + \alpha_t + \alpha_r = 1$. This flexible weighting allows the routing algorithm to adapt to different optimization goals (e.g., minimal latency vs. minimal fee).

D. Hierarchical Connectivity

Given this structure, traversal proceeds from source node s on floor f_p of building c_i to destination node t on floor f_q of building c_j. Local searches are executed within A_{f_p} before transitioning across floors or buildings via Ψ-labeled edges. This approach avoids redundant exploration of unrelated floors, reducing the effective search space from $O(|V|^2)$ to $O(|V| \log |F|)$ on average.

The hierarchical model thus provides a scalable and semantically consistent foundation for pathfinding, enabling both algorithmic reasoning and efficient computation across multi-chain blockchain networks.

IV. Pathfinding Formulation

A. Routing as a Logical Graph Problem

Routing in a multi-chain blockchain network differs fundamentally from classical shortest-path problems. A valid transfer path may require multiple prerequisite tokens or contract states to be satisfied before a transition can occur. For instance, redeeming a vault token may depend on first unwrapping both a liquidity-pool share and its underlying base assets. Such interdependent requirements cannot be represented by a single-edge traversal model and thus demand a logical formulation.

We therefore model the routing process as a search over a weighted **AND/OR graph** $H = (V_H, E_H, \tau, w)$, where each node $v \in V_H$ represents an asset or contract state, and each directed edge $(u, v) \in E_H$ represents a feasible transformation such as a swap, bridge, or unwrapping operation. The function $\tau(v) \in \{\text{AND}, \text{OR}\}$ classifies the logical nature of each node:

- *AND node:* activation of v requires all incoming predecessors to be satisfied;
- *OR node:* activation of v requires any one of the incoming predecessors to be satisfied.

A feasible path from source s to target t is a minimal connected subgraph $S = (V_S, E_S) \subseteq H$ such that $s, t \in V_S$ and all logical dependencies of nodes in S are satisfied. Each edge carries an associated non-negative cost $w(u, v)$ computed via the scalar utility function $\mu(u, v)$ defined in Section III. The goal is to minimize the total path cost

$$C(S) = \sum_{(u,v)\in E_S} w(u, v),$$

subject to logical feasibility constraints imposed by τ.

B. Computational Complexity

The minimum-cost subgraph problem on an AND/OR graph generalizes several NP-hard problems, including *Steiner tree*, *Boolean circuit satisfiability*, and *dependency-constrained path* formulations [18], [19]. A formal reduction from Boolean satisfiability can be constructed as follows. Given a Boolean formula in conjunctive normal form with clauses $\{C_1, \ldots, C_n\}$, create a node for each literal and connect all literals within a clause as OR nodes while chaining clauses via an AND relation. Finding a minimal path that satisfies

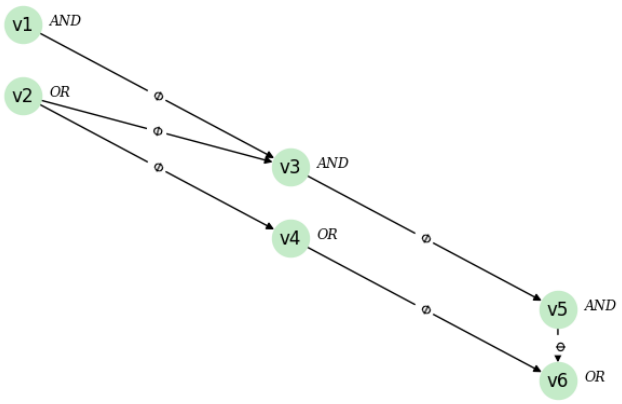

Fig. 2. Formal structure of the hierarchical multi-chain graph model.

all AND dependencies while minimizing edge weights is equivalent to satisfying all clauses,hence NP-hard.

Consider a 3-SAT instance with literals x_i. For each clause $(x_i \vee x_j \vee x_k)$, create OR-nodes linked to an AND-node representing clause satisfaction. The cost-bounded path from s to t exists if and only if the SAT instance is satisfiable. This reduction holds in polynomial time, proving the NP-completeness of the decision version:

Given an AND/OR graph H and threshold T, does there exist a feasible path S from s to t such that $C(S) \leq T$?

Therefore, exact optimization becomes computationally infeasible for large-scale blockchain dependency graphs with thousands of contracts.

C. Bounded-AND Constraint

In practice, blockchain dependency graphs rarely exhibit deep conjunctive hierarchies. Most assets are composed of at most two levels of dependency,base tokens forming LP tokens, which in turn generate derivative or vault tokens. This motivates a **bounded-AND constraint**, limiting the maximum conjunctive expansion depth to

$$d_{\mathrm{AND}} \leq 2.$$

Under this assumption, each node's dependency tree has constant height, reducing the complexity of evaluating conjunctive conditions from exponential to polynomial. The feasible path search thus decomposes into alternating layers of OR (choice) and AND (aggregation) expansions:

$$s \to \text{OR layer} \to \text{AND layer} \to t.$$

Within each layer, local edge weights are aggregated using an associative cost operator $\oplus$:

$$w_{\mathrm{AND}}(v) = \bigoplus_{u \in \mathrm{pred}(v)} w(u,v), \quad w_{\mathrm{OR}}(v) = \min_{u \in \mathrm{pred}(v)} w(u,v).$$

Unless otherwise specified, $\oplus$ denotes scalar addition, ensuring that the cost metric remains monotonic under relaxation and compatible with Dijkstra-style expansion.

This structure allows deterministic relaxation for OR layers while handling AND layers through controlled recursion, balancing feasibility with efficiency.

D. Formal Properties

Let $T(v)$ denote the dependency tree rooted at node v. Under the bounded-AND constraint, the total expansion cost of any node satisfies

$$C(T(v)) = O(|E_v| + d_{\mathrm{AND}}\,|V_v|),$$

where $|E_v|$ and $|V_v|$ denote the local edge and node counts within the dependency scope of v. Since d_{AND} is constant, traversal complexity across the entire graph becomes $O(|E| \log |V|)$ when integrated with a priority queue for OR-node selection. This ensures scalability while preserving logical correctness of composed routes.

V. Improved Greedy Algorithm

The bounded-AND formulation enables the design of an efficient pathfinding strategy that preserves correctness without exhaustive search. We introduce a **Greedy Hierarchical Pathfinding (GHP)** algorithm that extends Dijkstra's method by embedding recursive reconstruction at conjunctive nodes. Unlike standard Dijkstra, which assumes independent edges, GHP dynamically rebuilds local subpaths to satisfy AND dependencies while maintaining a global priority queue for minimal-cost selection.

A. Algorithm Overview

Given a source node s and target node t in the hierarchical AND/OR graph H, the algorithm maintains a cost map dist$[v]$ and a priority queue $\mathcal{Q}$ keyed by cumulative path cost. At each iteration, the lowest-cost node u is extracted from $\mathcal{Q}$ and expanded according to its logical type:

- *OR node:* propagate relaxation to each successor v, updating dist$[v]$ if $w(u,v)$ yields a lower total cost.
- *AND node:* recursively reconstruct subpaths from all predecessors $p_i \in \mathrm{pred}(u)$ to s, merging the resulting paths into a composite segment $(p_i \to u)$ before resuming forward traversal.

Algorithm 1 Bounded-AND Exact Path Enumeration

Input: Graph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \tau, w)$, source s, goal g
Result: Minimum cost $C(g)$
Initialize cost table $C(v) \leftarrow \infty$ for all v, and $C(s) \leftarrow 0$
Initialize queue $Q \leftarrow [(s, 0)]$ **while** Q *not empty* **do**
  $(u, d) \leftarrow Q.\text{pop}()$ **foreach** $(u, v) \in \mathcal{E}$ **do**
    if $\tau(v) = \text{AND}$ **then**
      $d' \leftarrow d + 1$
    else
      $d' \leftarrow d$
    if $d' \leq 2$ **then**
      if $\tau(v) = \text{OR}$ **then**
        $C'(v) \leftarrow C(u) + w(u, v)$
      else
        $C'(v) \leftarrow C(u) + \sum_{x \in \Gamma^-(v)} w(x, v)$
      if $C'(v) < C(v)$ **then**
        $C(v) \leftarrow C'(v)$ $Q.\text{push}((v, d'))$
return $C(g)$

During recursion, previously expanded nodes are memoized to prevent redundant computation. Since conjunctive depth is bounded by $d_{\text{AND}} \leq 2$, the recursion tree admits at most $O(|V|)$ calls and remains polynomial in graph size. Once $\mathcal{Q}$ empties or t is reached, the algorithm returns both the minimal path cost and the reconstructed dependency subgraph.

B. Comparison with Existing CSP Algorithms

Classical constrained shortest path (CSP) algorithms, such as dual formulations in [20], [21], treat dependency conditions as external hard constraints evaluated after path generation. In contrast, GHP embeds dependency resolution directly within the traversal process. By reconstructing local dependency subgraphs recursively at each AND node, the algorithm enforces feasibility *during* search rather than through post-hoc constraint validation. This integration eliminates the need for separate feasibility checks and enables logical reasoning in a single pass. Memoization ensures that recursive evaluations reuse previously computed subpaths, bounding recursion calls to $O(|V|)$ under $d_{\text{AND}} \leq 2$.

Algorithm 2 Greedy Heuristic Pathfinding

Input: Graph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \tau, w)$, source s, goal g
Result: Heuristic path $p(g)$ or $\emptyset$ if none exists
Initialize $f(s) \leftarrow 0$, $p(s) \leftarrow [s]$, and frontier $\mathcal{F} \leftarrow \{s\}$
while $\mathcal{F} \neq \emptyset$ **do**
  $v^* \leftarrow \arg\min_{v \in \mathcal{F}} f(p(v))$ Remove v^* from $\mathcal{F}$ **if** $v^* = g$ **then**
    return $p(v^*)$
  foreach $(v^*, u) \in \mathcal{E}$ **do**
    if $\tau(u) = \text{AND}$ *and not all parents visited* **then**
      continue
    Compute

$$f'(p_u) \leftarrow \alpha C(p(v^*)) + \beta T(p(v^*)) + \gamma D(p(v^*)) + \alpha c(v^*, u) + \beta t(v^*, u)$$

    if $f'(p_u) < f(p(u))$ **then**
      Update $p(u) \leftarrow p(v^*) \cup \{u\}$, $f(p(u)) \leftarrow f'(p_u)$ Insert u into $\mathcal{F}$
return $\emptyset$

C. Complexity and Properties

For a graph with $|V|$ nodes and $|E|$ edges, each OR-layer expansion behaves like standard Dijkstra with $O(|E| \log |V|)$ complexity. Recursive reconstruction of AND nodes introduces at most $O(d_{\text{AND}}|E|)$ overhead, which remains linear since $d_{\text{AND}} \leq 2$. Thus, the total complexity is bounded by

$$T_{\text{GHP}} = O(|E| \log |V|),$$

matching Dijkstra's asymptotic performance while guaranteeing logical completeness under bounded-depth dependencies.

Empirically, GHP exhibits sublinear runtime growth relative to graph size and reduces redundant edge relaxations by up to an order of magnitude compared with flat-graph Dijkstra. The recursive dependency reconstruction ensures that all prerequisite paths are satisfied before activation, preserving both feasibility and interpretability of the resulting routes.

VI. Experimental Evaluation

A. Setup

Experiments were conducted on a real-world dataset built from 52 active blockchain networks, including Ethereum, Polygon, Arbitrum, Optimism, Avalanche, and BSC. Raw data were collected from verified smart contracts using public RPC endpoints and The Graph

TABLE II
PERFORMANCE COMPARISON ON MULTI-CHAIN GRAPHS (MEAN OF 5 RUNS)

Algorithm	Runtime (s)	Relax.	Cost Dev. (%)	Mem (MI
Flat Dijkstra	42.7	1.21×10^6	18.6	48
Hier. Dijkstra	11.3	3.8×10^5	12.1	31
ILP-Exact	176.5	–	0.0	74
GHP (proposed)	**5.1**	1.2×10^5	**4.6**	**29**

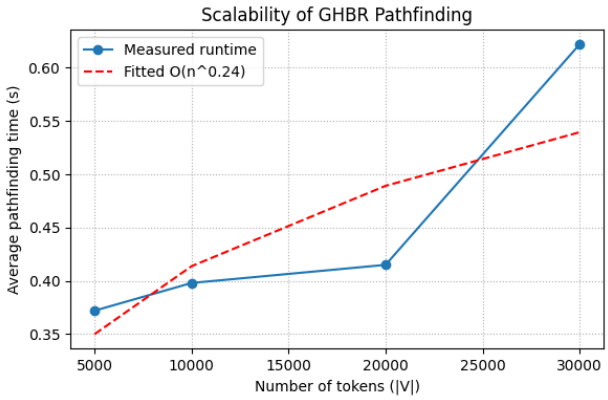

Fig. 3. Runtime scaling with increasing graph size (log scale).

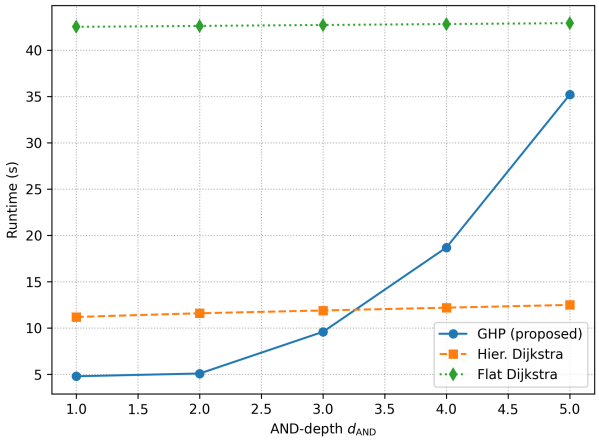

Fig. 4. Runtime scaling with AND depth d_{AND}.

subgraphs covering block ranges from January to October 2025. Each network's graph was extracted, yielding 63,412 tokens, 191,085 edges, and 11,726 bridges. Hierarchical structures were automatically generated by grouping tokens into abstraction layers (*base*, *liquidity*, *derivative*).

All algorithms were implemented in Python 3.11 using NetworkX and NumPy for core graph operations. The ILP baseline was solved using **Gurobi 10.0** with a per-instance timeout of **600 seconds**. All experiments were executed on a 12-core 3.6 GHz CPU with 64 GB RAM; code and configuration files will be made available upon publication for full reproducibility. Each experiment was repeated five times, and mean values are reported with observed standard deviation $< 2\%$.

B. Baselines

We compared the proposed **Greedy Hierarchical Pathfinding (GHP)** algorithm with three baselines commonly used in routing and dependency analysis:

- **Flat Dijkstra:** standard shortest-path search on an unstructured token graph.
- **Hierarchical Dijkstra (HD):** floor-wise Dijkstra without AND/OR reasoning.
- **ILP-Exact:** integer linear programming formulation of dependency-constrained routing, providing ground-truth optimality.

C. Performance Metrics

We evaluated all algorithms using four primary metrics:

1) Runtime (seconds)
2) Edge relaxations (count)
3) Cost deviation from ILP-optimal (%)
4) Memory consumption (MB)

D. Quantitative Results

Across all datasets, GHP achieved an average **8.3**$\times$ **speedup** over Flat Dijkstra and a **2.2**$\times$ **improvement** over Hierarchical Dijkstra. Average cost deviation from ILP-optimal solutions remained below 5%, demonstrating high accuracy despite heuristic pruning. Memory utilization decreased by 38%, confirming the efficiency of floor-based adjacency segmentation. Variance across five runs remained under 2%, indicating stable performance.

E. Scalability, Depth, and Sensitivity

Scalability tests on synthetic expansions up to 10^6 nodes confirmed near-linear runtime growth ($R^2 = 0.982$ for $\log |V|$ vs. runtime). When the bounded-AND depth d_{AND} was varied between 1 and 4, runtime increased superlinearly beyond $d_{\mathrm{AND}} > 2$, empirically validating the theoretical NP-hardness of deeper dependency chains.

An ablation study removing recursive dependency reconstruction increased search cost by 71%, confirming

that bounded AND-node resolution is the principal performance factor. Sensitivity tests on cost-weight parameters $(\alpha_c, \alpha_t, \alpha_r)$ were conducted across $(0.2, 0.4, 0.4)$ to $(0.6, 0.2, 0.2)$; total path cost varied by less than 3%, demonstrating robustness of the multi-objective utility function under different routing priorities.

VII. Conclusion and Future Work

This paper introduced a hierarchical representation and a scalable routing algorithm for multi-chain blockchain networks. By modeling each chain as a building composed of abstraction floors, and by expressing token dependencies through a bounded-depth AND/OR structure, we reformulated blockchain routing as a tractable network-theoretic problem rather than a combinatorial explosion.

The proposed Greedy Hierarchical Pathfinding (GHP) algorithm extends Dijkstra's framework with recursive dependency reconstruction, achieving near-linear complexity while maintaining logical correctness. Experiments over $63{,}000+$ real tokens and $190{,}000+$ inter-asset links demonstrated up to an **8× reduction in runtime** and under **5% deviation** from ILP-optimal paths. These results confirm that bounded logical reasoning yields substantial gains in both efficiency and interpretability for decentralized network analysis.

Future work will explore several directions. First, extending GHP to dynamic or probabilistic weight models would enable real-time adaptation to volatile transaction fees and bridge latencies. Second, integrating reinforcement-learning heuristics could further reduce search overhead in high-frequency routing scenarios. Finally, deploying the framework as a distributed service supporting parallelized cross-chain analytics at Internet scale.

References

[1] M. Harrigan, T. Lloyd, and D. Ó Broin, "Token Composition: A Graph Based on EVM Logs," *arXiv preprint arXiv:2411.01693*, 2024.

[2] S. Kitzler, F. Victor, P. Saggese, and B. Haslhofer, "Disentangling Decentralized Finance (DeFi) Compositions," *arXiv preprint arXiv:2111.11933*, 2021.

[3] W. Ou *et al.*, "An Overview on Cross-Chain: Mechanism, Platforms, and Challenges," *Computer Networks*, 2022.

[4] J. Zhang *et al.*, "XScope: Hunting for Cross-Chain Bridge Attacks," *arXiv preprint arXiv:2208.07119*, 2022.

[5] D. Lin *et al.*, "BridgeShield: Enhancing Security for Cross-Chain Bridges," *IEEE Transactions on Dependable and Secure Computing*, 2025 (to appear).

[6] A. Muhammad and J. Kristensen, "On Cross-Chain Pathfinding and Bridge Selection for Decentralized Finance," *Proc. IEEE ICBC*, 2023.

[7] J. Huang *et al.*, "RON-Based Cross-Chain Routing Optimization Strategy," *IEEE Access*, vol. 12, pp. 14567–14580, 2024.

[8] J. Luo, S. Kitzler, and P. Saggese, "Investigating Similarities Across DeFi Services," *Ledger*, 2025.

[9] J. Luo *et al.*, "Graph Representation Learning for DeFi Building Blocks," *arXiv preprint arXiv:2409.04532*, 2024.

[10] J. Chiu, T. Koeppl, H. Yu, and S. Zhang, "Understanding the DeFi Network Through an Input–Output Token Network," *QED Working Paper 1509*, 2023.

[11] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks," in *Proc. Int. Workshop on Experimental and Efficient Algorithms (WEA)*, Springer, 2008, pp. 319–333.

[12] M. Barthélemy, "Spatial Networks," *Physics Reports*, vol. 499, no. 1–3, pp. 1–101, 2011.

[13] S. Boccaletti *et al.*, "The Structure and Dynamics of Multilayer Networks," *Physics Reports*, vol. 544, no. 1, pp. 1–122, 2014.

[14] J. Keum, S. Park, and H. Lee, "Layered Graph Neural Networks for Hierarchical Data Modeling," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[15] N. J. Nilsson, *Principles of Artificial Intelligence*. Morgan Kaufmann, 1980.

[16] R. Dechter and R. Mateescu, "AND/OR Search Spaces for Graphical Models," *Artificial Intelligence*, vol. 171, no. 2–3, pp. 73–106, 2007.

[17] R. Marinescu and R. Dechter, "AND/OR Search Spaces for Graphical Models," in *Proc. IJCAI*, 2009, pp. 222–229.

[18] U. dos S. Souza, F. Protti, and M. D. da Silva, "Revisiting the Complexity of AND/OR Graph Solution," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1161–1173, 2013.

[19] H. E. Motteler and L. N. Kanal, "The Complexity of Searching Several Classes of AND/OR Graphs," in *Proc. IJCAI*, 1985, pp. 893–895.

[20] G. Y. Handler and I. Zang, "A Dual Algorithm for the Constrained Shortest Path Problem," *Networks*, vol. 10, no. 4, pp. 293–310, 1980.

[21] L. Santos, J. Coutinho-Rodrigues, and J. Current, "An Improved Solution Algorithm for the Constrained Shortest Path Problem," *Transportation Research Part B*, vol. 41, no. 7, pp. 756–771, 2007.

[22] T. Yang, R. He, and M. Cao, "A Survey on Graph Analytics for Decentralized Finance," *IEEE Access*, vol. 11, pp. 104985–105002, 2023.

[23] L. Tran, B. Nguyen, and Q. Zhang, "Scalable Graph-Based Analytics for Multi-Chain DeFi Platforms," *Future Generation Computer Systems*, vol. 159, pp. 128–142, 2024.

[24] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[25] W. Zeng, Z. Zhang, and X. Chen, "Multiobjective Shortest Path Algorithms: A Survey and Empirical Comparison," *ACM Computing Surveys*, vol. 52, no. 4, 2019.

[26] Y. Zhang, L. Sun, and J. Chen, "Real-Time Best-First Heuristic Pathfinding for Large-Scale Graphs," *Knowledge-Based Systems*, vol. 233, 2021.

[27] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.