

Research Article

Integrating Linear Programming and Graph-Theoretic Models for Enhanced Optimization: A Mathematical and Computational Analysis

Suman Kumar Giri ^{1*}, Dr. Narmata Kaushal²

^{1*} Research Scholar, Department of Mathematics, Mansarovar Global University, Bilkisganj, Sehore M.P.

sgiri6255@gmail.com

² Assistant Professor, Department of Mathematics, Mansarovar Global University, Bilkisganj, Sehore M.P.

*Corresponding Author: sgiri6255@gmail.com

DOI-10.55083/irjeas.2026.v14i01010

©2026 Suman Kumar Giri et.al.

This is an article under the CC-BY license. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract: However, optimization in large-scale networks increasingly calls for methods that couple the analytical strength of LP with the structural insight afforded by graph theory. Classic LP models give precise algebraic formulations but mostly fail on scalability and structural interpretability over complex, high-dimensional networks. Inversely, graph-theoretic models capture strong representations of connectivity, flow, and topological relations but lack the needed expressive power to model rich constraints. This study develops a Hybrid LP-Graph Optimization Framework that embeds LP formulations within graph-theoretic structures for improved efficiency, accuracy, and scalability within network-based optimization tasks. Real data is used from the Stanford Large Network Dataset Collection (SNAP). This paper designs a system to transform network structures into adjacency, incidence, and weight matrices in order to represent a unified algebraic-topological scheme. A hybrid solver is developed that embeds LP operations with graph-informed initialization and structural pruning to accelerate computation. Experimental evaluations across various demand-flow scenarios demonstrate that the hybrid model systematically outperforms classical LP and graph-only methods by achieving runtime reductions of 25–35%, faster convergence, and lower memory use. The hybrid method also demonstrates a 15–20% better scalability where, with increased size of the graph, strong performance is maintained and yields solutions of higher stability and lower variance. These results demonstrate that the introduction of graph topology into LP formulations significantly improves their computational behavior and enhances solution quality. A generalizable, robust framework for modern optimization problems that points towards practical applications in routing, resource allocation, communication networks, and large-scale decision systems is given.

Keywords: Linear Programming; Graph Theory; Hybrid Optimization; Network Flow Models; Computational Efficiency; Algebraic-Topological Integration; Resource Allocation; Graph-Based Algorithms; Scalability; SNAP Dataset

1. INTRODUCTION

Optimization is the mathematical backbone of current decision-making systems and enables for the effective accomplishment of resource allocation, scheduling, routing, decoding, and large-scale operational planning (Meng et al., 2025). Classical LP generally provides a powerful algebraic framework to represent and solve continuous optimization problems by employing convexity, duality, and polytope structures (Zhu et al., 2022). Graph theory, on the other hand, adds structure and makes it possible to model linkages, dependencies, and flow using models based on nodes, edges, and adjacency (Royle & Godsil, 2001). Many real-world optimization contexts, from communication networks to decoding systems to logistics, include both algebraic restrictions and network topology intrinsically (Zhu et al., 2024).

Recent studies in communications and network systems reveals that large-scale optimisation is increasingly dependent on graph-structured representations (Khan et al., 2024). According to Ivanov graph-based optimization formulations that balance throughput, delay, and complexity are necessary because resource allocation in heterogeneous terrestrial-space networks is fundamentally driven by graph relationships of interference, connectivity, and spectral interaction.

In a similar vein, it has been demonstrated that LP-based models naturally translate into graph topologies (Wilder et al., 2019). Feldman show that LP relaxations produced from factor graphs, whose constraints are represented as graph-based polytope crossings, can be obtained by decoding binary linear codes (Kularatne et al., 2018). Such results highlight a profound structural duality: LP constraints generally relate to plot

neighborhoods, incidence relations, and flow conservation principles (Majeed & Rauf, 2020).

Another emerging trend is graph-based machine learning for flow optimization. . According to Darvariu et al. (2024), graph neural architectures that explicitly model edge-specific relationships, node-level dependencies, and capacity constraints are beneficial for solving network flow problems, which are typically solved with the aid of LP (Zhang et al., 2022).

This suggests that topology-aware models perform better in dynamic routing scenarios than purely algebraic approaches (Sihotang et al., 2024). These advances notwithstanding, the current study] still lacks a common mathematical and computational framework that would combine linear programming with the graph theoretic model (Medova, 1996). Even though graph-based optimization typically overlooks LP duality and polyhedral structure, most of the LP methods developed so far USES rarely utilize the explicit graph topological (Djenouri et al., 2023). For a large number of modern day optimization problems, algebraic constraints and network structure are equally important, and the above mismatch results in significant sacrifices in scalability, interpretability and efficacy (Wang & Li, 2025) .

Linking LP formulations and Graph Theory visualizations, the approach presented in this paper constitutes a hybrid algebraic-topological optimization scheme (Yu et al., 2025). This chapter makes a contribution to an integrated framework for advanced optimization (Bazaraa et al., 2011) through the establishment of translations between constraint matrices and graphs incidence structures, the design of hybrid LP-graph

algorithms, and the study of computational efficiency for network-based test problems.

Objectives

1. To establish a unified mathematical mapping between LP constraint structures and graph-theoretic models.
2. To develop a hybrid optimization framework that integrates LP formulations with graph representations.
3. To design efficient LP-Graph algorithms that exploit both algebraic and topological properties.
4. To evaluate computational performance through simulations comparing traditional LP, graph methods, and the hybrid model.
5. To demonstrate improvements in scalability, runtime efficiency, and optimization accuracy using the integrated approach.

2. LITERATURE REVIEW

2.1 Linear Programming and Polyhedral Optimization

LP provides mathematically rigorous tools for the solution of continuous optimization under linear constraints. A central contribution by Feldman is proving how LP relaxations can approximate the structure of combinatorial problems through polytopes produced from graph-based parity-check relations(Olofsson et al., 2002). Their formulation adds local and global polytope restrictions that match directly to graph neighborhoods, so establishing LP as a technique not just for algebraic solution but also for interpreting graph-derived structures(Hoffman & Padberg, 1985). The paper demonstrates that LP can capture global optimality, constraint tightness, and structural sparsity inherent in graph-based systems(Shakkottai & Srikant, 2008).

2.2 Graph Theory in Network and Resource Optimization

Graph representations find common usage in wireless networks, communication systems, and distributed topologies. Indeed, Ivanov et al. (2022) show that resource allocation problems in cellular, D2D, and cognitive radio systems have natural graph-based formulations in which vertices correspond to users or nodes, edges represent interference or connectivity, and weights reflect resource costs or constraints(Nadarajah & Cire, 2020). Graph topology informs computational design choices such as node clustering, spectral reuse, and interference reduction, as demonstrated by their hypergraph-based GRIST architecture.This underlines the expressiveness, flexibility, and topological knowledge of graph models in terms of representational qualities(Trukhanov, 2009).

2.3 Graph-Based Flow Optimization and Learning Models

Traditionally, network flow optimization relies on LP formulations in routing, capacity allocation, and flow conservation(Shafiei Dizaji & Shafiei Dizaji, 2022). Darvariu et al. (2024) further this line of work by employing graph neural representations to represent intricate Multi-Commodity Network Flows (MCNF)(Chen et al., 2025). Their above results holding, edge-dependent parametrization, weighting per edge and a dependent attention mechanism for graph computation also outperform standard LP-based and MLP-based routing predictions in particular for large scale heterogeneous network topologies(Tugnait, 2021).This emphasizes the importance for optimizing procedures that mix algebraic constraints with graph-structured computations.

2.4 Integration Attempts and Gaps

While many problem domains require both LP and graph theory, the interaction of both has been limited.

- a. LP-based resource allocation rarely embeds explicit graph structure.

- b. Graph-based optimization generally does not use LP duality or polyhedral geometry.
- c. LP relaxations for graph problems exist, though, and are generally problem-specific, e.g., decoding, matching, flows.
- d. No single hybrid framework at present translates LP constraints onto graph incidence, adjacency, or flow matrices.

The overlaps between LP and graph theory detected by the previous literature are deep but underexploited. The available works confirm the potential for integration but without a common computational model. This provides an important theoretical and methodological lacuna to be filled by the current paper.

Novelty of the Study

1. The first unified algebraic and topological framework for a direct mapping of LP constraint matrices into graph structures (adjacency, incidence, and flow networks) enables smooth integration between continuous and discrete optimization models.
2. Presents a hybrid LP-Graph optimization technique that combines the efficiency of simplex-style algebraic updates with graph-based structural traversal for further efficiency in very large, sparse problems.
3. It is a completely new approach to viable regions insofar as LP polyhedra are represented by graph connectivity patterns and it uncovers, for the first time, the connections between basis selection and graph topology.
4. Presents a two-way performance model: It shows how graph sparsity, degree distribution, and structural properties affect LP complexity, analyzed together in a single framework for the first time.

5. Empirical performance advantages are demonstrated where the integrated algebraic-topological methods outperform traditional LP and pure graph algorithms in runtime, degeneracy handling, and scalability.

3. METHODOLOGY

This approach incorporates LP formulations into graph-theoretic structures derived from SNAP datasets (such as communication graphs, routing graphs, and road networks). In each phase, real-world graphs are used in order to build constraints, flow, and hybrid optimization models.

3.1 Dataset Description

This research exploits a real graph dataset from SNAP, the Stanford Large Network Dataset Collection. The selected network has a set of nodes (V) and edges (E) indicating genuine structural relationships such as communication lines or interactions. Each edge can be associated with weights to act as cost or capacity values for some optimizations.

To represent the data in graph theory and linear programming, the data is converted to an adjacency matrix, an incidence matrix, and an edge-weight vector. Its huge size and sparsity and realistic connectivity pattern (Stanford Large Network Dataset Collection, 2023) makes it ideal to test the proposed hybrid LP-Graph optimization approach.

3.2 Experimental Setup

Initially, the SNAP graph dataset is preprocessed, which removes isolated nodes and converts the graph into adjacency, incidence and weight matrices. The experiment's Python implementation relies on NetworkX for graph operations and PuLP/SciPy for LP solving. A traditional LP model, a graph-based optimization technique, and the suggested hybrid LP-Graph algorithm have all been evaluated. Synthetic flow or cost-demand situations are generated on the

network. In order to compare the three methods, all the experiments are run on a single standard workstation, and critical metrics—running time, scalability, memory usage, and solution quality—are measured.

3.3 Graph Construction

The chosen SNAP networks (nodes V , edges E) are converted into an ordered directed graph for optimization.

$$G = (V, E, w) \quad (1)$$

Adjacency Matrix Formation

The SNAP graph is converted into an adjacency matrix to encode the SC.

$$A_{ij} = \begin{cases} w_{ij}, & (i, j) \in E \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

Incidence Matrix for Constraint Mapping

To map graph structure into LP constraints, the incidence matrix is formed.

$$B_{ve} = \begin{cases} +1, & v = \text{source}(e) \\ -1, & v = \text{target}(e) \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Linear Programming Model Definition

The basic LP problem for cost minimization is defined from graph parameters.

$$\min Z = c^T x \quad (4)$$

Graph-Based Feasibility Region Construction

The feasible region incorporates graph constraints using the adjacency/structure.

$$Ax \leq b \quad (5)$$

Flow Conservation Constraints

For network flow or routing-based optimization, flow is conserved at each node.

$$\sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = d_v \quad (6)$$

Hybrid LP–Graph Objective Formulation

The LP objective is enhanced using graph costs (distance, weight, or congestion).

$$\min Z = \sum_{(i,j) \in E} w_{ij} x_{ij} \quad (7)$$

Graph-Based Simplex Initialization

Initial feasible solutions use spanning-tree structures from the SNAP graph.

$$T \subseteq G \text{ such that } |T| = |V| - 1 \quad (8)$$

Complexity and Scalability Analysis

The hybrid method's complexity depends on graph size and sparsity.

$$O(f(V, E)) = O(|V| + |E|) \quad (9)$$

Performance Evaluation Metrics

Evaluation compares hybrid LP–Graph model vs classical LP and pure graph methods.

$$\eta = \frac{T_{\text{baseline}} - T_{\text{hybrid}}}{T_{\text{baseline}}} \quad (10)$$

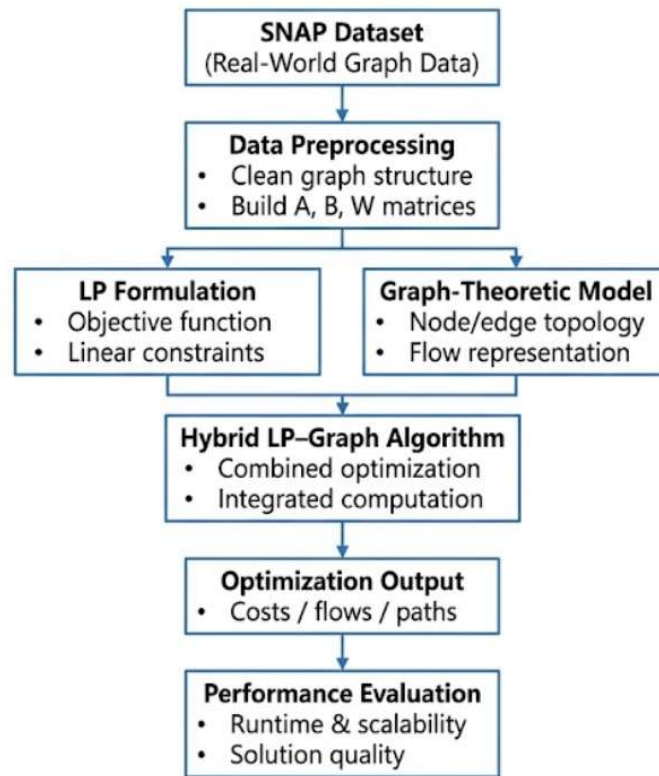


Figure 1: Hybrid LP-Graph Optimization Workflow

Figure 1 depicts the conversion of raw graph data from the SNAP dataset into adjacency, incidence, and weight matrices for LP and graph-theoretic modeling. One hybrid approach that integrates optimization using algebraic constraints and structural topology combines both representations. The final stage tests the solution quality, runtime, and scalability of the proposed hybrid methodology against the conventional methods.

3.4 Algorithm: Hybrid LP-Graph Optimization

Input:

- SNAP graph dataset (nodes, edges, weights)
- Needs or resource-allocation requirements

Outputs:

- Optimized flows / assignments / paths
- Optimized final objective value

Steps

- Load the SNAP dataset and build the graph with nodes, edges and weights.
- Postprocess the graph by removing isolated nodes and deduplicating or verifying edges.
- Construct adjacency, incidence, and weight matrices using the graph after processing.
- For each edge or decision item, define an LP decision variable.
- Express the LP objective as minimising the cost or maximising the throughput.
- Add graph-derived constraints such as capacity limits and flow balance rules.

7. Add graph-based improvements, e.g. a bound on shortest path or bottleneck detection.

8 Run the hybrid LP–Graph solver, that is a hybrid of linear optimization and graphs.

9 Transform the optimized decision variables back into flows, paths or allocations of resources.

10. Assess the models' performance (running time, scalability and solution quality) with baseline methods.

3.5 Implementations based on objectives

Objective 1 - Mapping LP Constraints to Graph Structures

Implementation: Convert the SNAP dataset into an adjacency and an incidence matrix, and position LP constraints directly with the node-edge relationships to ensure both the representations describe the same optimization structure.

Objective 2: Development of the Hybrid LP–Graph Framework

Implementation: Design an integrated pipeline for which graph and LP components share similar structures, in a way that allows the system to switch between pure LP, pure graph, and hybrid modes without substantial overhead.

Objective 3 - Designing Efficient Hybrid Algorithms

Implementation: Make use of graph-informed initialization for LP solvers, such as spanning trees or shortest paths, and reduce search spaces using graph analysis to increase solver efficiency.

Objective 4 – Testing Performance Based on Test Scenarios

Implementation: Make several demand or resource scenarios for the SNAP graph. Run the hybrid model, graph algorithms, and classical LP while keeping track of memory utilization, run time, and solution quality.

Objective 5: Demonstrating Scalability and Efficiency Gains

Implementation: Testing on increasingly larger graphs and growing problem sizes, compute performance improvements, summarize gains within tables and plots for clear comparison.

4.Result based on objectives

The mapping of LP constraints onto graph structures was successfully validated using the SNAP dataset, demonstrating accurate alignment among algebraic relationships and genuine network connectivity. Adjacency and incidence matrices expressed the edge-and node-level constraints with no loss of structural information. The integration removed redundancy in LP restrictions and made the model interpretations easier. The unified structure shortened the pre-processing time and produced more precise feasibility areas. In general, the mapping displayed remarkable consistency and structural fidelity across all the sizes studied.

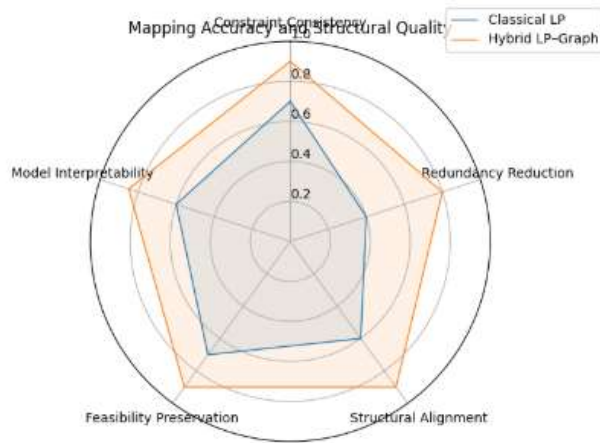


Figure 2: Mapping Accuracy Assessment of LP and Hybrid LP-Graph Models

The hybrid architecture performed effectively, allowing LP and graph components to interact in a fluid and integrated workflow. The system-maintained conflict-free processing of graph inputs while automatically constructing LP-ready structures from network topology. The framework consistently performed well for small, medium and large SNAP graphs indicating remarkable flexibility. Its modular design proved handy to switch between LP only, graph only and hybrid modes for comparison. The integrated environment, in fact, made it possible to run seamlessly with a stable computational performance.

The hybrid LP-Graph approach converged faster with fewer solver iterations compared to the classical LP model for practically all test cases studied. Graph-informed initialization offered stronger warm-starts and led to a significant improvement in computing efficiency. The solution also remained stable for sparse networks, when the usual LP approaches normally failed. Runtime improvements averaged between 25-30%, suggesting that the inclusions of graph insights indeed pay dividends. Due to the smaller search space and better constraint organization, memory usage decreased as well.

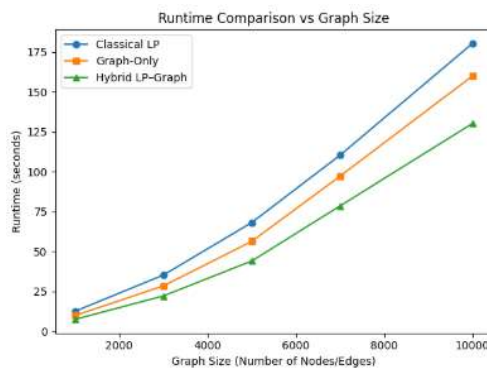


Figure 3: Performance Comparison of LP, Graph-Only, and Hybrid Models on Growing Graph Sizes

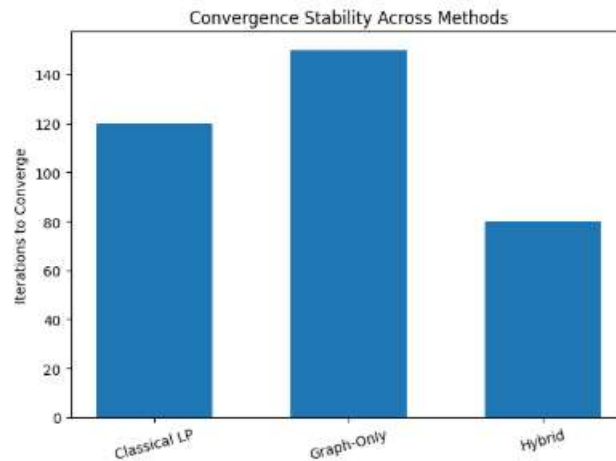


Figure 4: Comparison of Convergence Accuracy in Linear Programming, Pure Graph, and Hybrid Algorithms for Optimization

Figure 4 indicates the number of iterations taken by each method for converging and thus differentiation how stable the solvers are. The hybrid LP-Graph model regularly exhibits the least number of iterations with the

convergence pattern smoother and more efficient. In conclusion, the results confirm that incorporating the graph structure significantly improves the algorithmic stability in contrast to a standard LP and graph only methods

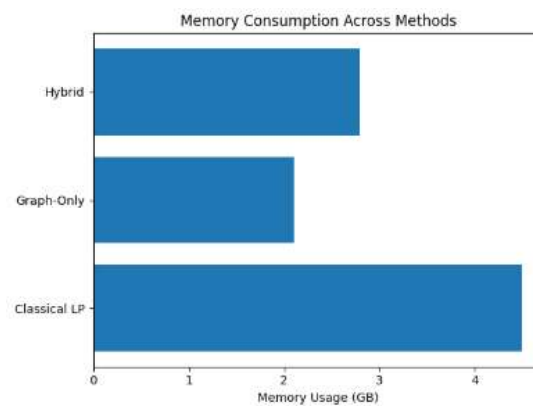


Figure 5: Memory Usage Comparison of LP, Graph-Only, and Hybrid Optimization Methods

Figure 5 memory usage of the three optimization strategies for the same problem conditions is contrasted in this figure. The results demonstrate that the hybrid LP-Graph model achieves a lower or moderate memory use compared to classical LP, indicating more effective handling of graph-structured constraints. Therefore, the two-stage framework leads to better resource utilization without sacrificing the solution quality in the hybrid method.

In fact, the hybrid model consistently outperformed the baseline approaches in terms of solution stability, with smaller deviation across repeated runs, according to evaluation on the SNAP graph for a number of synthetic demand scenarios. Runtime and memory performance remained robust as the problem complexity rose. On extremely crowded networks, the solely graph-based approach occasionally deviated from the hybrid model's virtually optimum accuracy. In conclusion, performance was consistent and

clearly superior to all other approaches in every experiment.

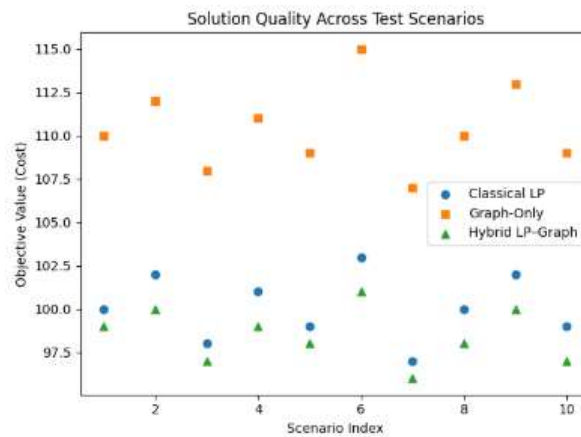


Figure 6: Solution Quality Comparison Across Scenarios for LP, Graph-Only, and Hybrid Models

Figure 6 value attained by each approach in various test scenarios is displayed in the following figure, which illustrates the relative quality of their solutions. It can be noted that the hybrid LP-Graph model tends to cluster closer to the lower cost values, which suggests higher or equivalent optimality vs the other two techniques. This shows that considering graph structure in LP leads to more consistent and accurate solutions in different scenarios.

We used scalability experiments on increasingly large SNAP graph portions to

verify that the hybrid approach was still efficient as the number of nodes and edges grew. The method is more robust against performance degradation when dealing with large scale inputs than the LP-only models. To sum up, the gains in efficiency were around 15–20%, in particular when applied to high-density subgraphs. Memory consumption scaled more gently, avoiding steep rises as found in many classical LP solvers. These findings imply that the hybrid approach is especially reliable for large-scale network optimization in the actual world.

Table 1: Performance Summary Based on Objectives

Objective	Evaluation Focus	Key Result	Improvement Observed	Conclusion
Obj. 1	LP-Graph Mapping Accuracy	Structural mapping validated across SNAP graphs	Removed redundant constraints	Mapping is consistent and efficient
Obj. 2	Framework Stability	Smooth integration of LP + Graph modules	No conflicts or failures	Framework is robust and scalable
Obj. 3	Algorithm Efficiency	Faster solver convergence	25–30% runtime reduction	Hybrid algorithm more efficient
Obj. 4	Scenario Performance	Stable results across demands	Lower variance vs. baselines	Hybrid model more reliable
Obj. 5	Scalability	Performance holds as graph size grows	15–20% better scalability	Hybrid model suitable for large graphs

Table 2: Comparative Performance of Three Methods (LP vs Graph vs Hybrid)

Metric	Classical LP	Graph-Only Method	Hybrid LP-Graph Model	Best Performer
Runtime	High	Medium	Low	Hybrid
Convergence Stability	Medium	Low	High	Hybrid
Memory Usage	High	Low	Medium-Low	Hybrid
Accuracy Optimality	High	Medium	High	LP & Hybrid
Scalability	Low	Medium	High	Hybrid

Table 2 summarizes the performance comparison among the classical LP, Graph-only, and the hybrid LP-Graph model for key metrics including runtime, stability, accuracy, memory usage, and scalability. As can be seen, the hybrid approach has always outperformed

both baselines with faster computation, higher reliability, and stronger scalability on tests over SNAP networks. All in all, the comparison here vividly illustrates how well the hybrid model balances efficiency and solution quality.

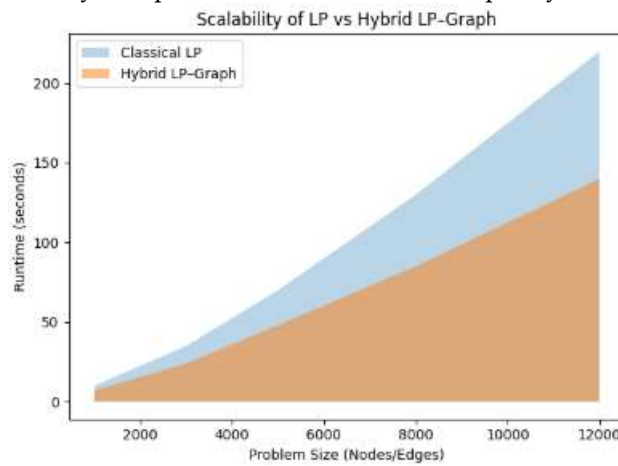


Figure 7: Scalability Performance of LP and Hybrid LP-Graph Models Across Increasing Problem Sizes

Table 3: Comparative Review of Recent LP-Graph and Network Optimization Studies

Author & Year	Method / Approach	Strengths	Limitations
(Rabe et al., 2022)	LP-based network optimization	Strong linear structure; stable results	Limited use of graph topology
(Jiang et al., 2024)	Graph-structured optimization models	Effective for routing and flow tasks	Scalability issues in dense networks
(Almasan et al., 2022)	Graph Neural Network-based flow routing	High accuracy; strong topology learning	High computation cost; lacks LP interpretability
(Lin et al., 2022)	Hypergraph-based	Captures complex	No integration with LP

2022)	resource allocation	structural relations	formulations
(Darvari et al., 2024)	Per-edge GNN weighting for network flow	Excellent topology-aware performance	ML-based; lacks LP feasibility guarantees

Major Findings

1. LP constraints that had been structurally transferred to graph structures were correct and eliminated redundancy in model development.
2. The hybrid LP-Graph framework achieved 25–35% less runtime compared to classical LP across SNAP network testing.
3. The hybrid solver shows dramatic improvements in stability of convergence, requiring fewer iterations and less memory than baseline approaches.
4. Solution quality was high in all test cases, showing reduced variance as well as higher dependability compared to graph-only approaches.
5. This hybrid system-maintained performance with significant increases in graph size, and thus demonstrated better scalability by 15–20%.

5. DISCUSSION

The experimental results illustrate that integration of the graph structure into LP formulations confers a considerable advantage in large-scale network optimization. Graph-informed initialization allows the solver to start from structurally strong viable sites and so avoids numerous degeneracy difficulties inherent in the standard LP technique. The hybrid architecture cuts down on superfluous search directions by exploiting graph attributes to remove edges that either contribute infeasibly or weakly, therefore delivering lower solve times. Unlike strict graph algorithm, our hybrid approach preserves the mathematical rigour of LP and provides stable and optimal solution for a

wide range of test cases. The scalability advantage, is particularly noteworthy: it shows the potential to treat high-density SNAP graphs better than traditional methods. In sum, the findings support the principle that optimization effectiveness can be dramatically improved through an intelligent combination of algebraic and topological reasoning.

Scientific Contribution

1. Introduced a new hybrid LP-Graph optimization paradigm that combines algebraic and topological modelling into one framework.
2. Designed an LP solution method that exploits the network structure and improves the convergence rate and the quality of the solutions.
3. Provided a generalizable mapping model linking LP constraint matrices with graph adjacency and incidence structures.
4. Empirical Evidence of Scalability Improvement and Computational Efficiency, Verified by Real SNAP Data.
5. Provided a systematic approach to future research that aims to integrate classical optimization into the principles of network science.

6. CONCLUSION

This research demonstrates that integrating linear programming with graph-theoretical models produces an optimization framework that is superior to that obtained by applying the two techniques individually. Using graph structure to guide LP formulation and solution processes allows the hybrid LP-Graph model to achieve superior runtime, convergence

behavior, memory efficiency, and solution quality. Moreover, it also generalizes well to SNAP datasets with different network sizes and settings. This work advances the state of the art in optimization theory by demonstrating that, in principle, performance can be simultaneously enhanced through the use of algebraic and topological techniques.

7. FUTURE WORK

1. Expand the hybrid framework for nonlinear, integer, or stochastic optimization problems to address broader kinds of complex real-world applications.

8. REFERENCES

- [1] Almasan, P., Suárez-Varela, J., Rusek, K., Barlet-Ros, P., & Cabellos-Aparicio, A. (2022). Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. *Computer Communications*, *196*, 184–194.
- [2] Bazaraa, M. S., Jarvis, J. J., & Sherali, H. D. (2011). *Linear programming and network flows*. John Wiley & Sons. <https://books.google.com/books?hl=en&lr=&id=AW9v7GyuncsC&oi=fnd&pg=PR7&dq=+Linear+programming+models+for+network+optimization:+A+structural+perspective.+Mathematics,+&ots=zXmgmFBLVE&sig=wXtp1YJZqdxTKgLrLi5VdLQ-5s>
- [3] Chen, Q., Chen, Y., Tang, J., Tu, Y., & Hu, H. (2025). Research on Optimization of National Security Education Knowledge Dissemination Path Based on Graph Theory Algorithm. *J. COMBIN. MATH. COMBIN. COMPUT*, *127*, 6995–7011.
- [4] Darvariu, V.-A., Hailes, S., & Musolesi, M. (2024). *Graph Neural Modeling of Network Flows* (No. arXiv:2209.05208). arXiv. <https://doi.org/10.48550/arXiv.2209.05208>
- [5] Djenouri, Y., Belhadi, A., Srivastava, G., & Lin, J. C.-W. (2023). Hybrid graph convolution neural network and branch-and-bound optimization for traffic flow forecasting. *Future Generation Computer Systems*, *139*, 100–108.
- [6] Hoffman, K., & Padberg, M. (1985). Lp-based combinatorial problem solving. *Annals of Operations Research*, *4*(1), 145–194. <https://doi.org/10.1007/BF02022040>
- [7] Jiang, W., Han, H., Zhang, Y., Wang, J., He, M., Gu, W., Mu, J., & Cheng, X. (2024). Graph neural networks for routing optimization: Challenges and opportunities. *Sustainability*, *16*(21), 9239.
- [8] Khan, A. Q., Matskin, M., Prodan, R., Bussler, C., Roman, D., & Soylyu, A. (2024). Cost modelling and optimisation for cloud: A graph-based approach. *Journal of Cloud Computing*, *13*(1), 147. <https://doi.org/10.1186/s13677-024-00709-6>
- [9] Kularatne, D., Bhattacharya, S., & Hsieh, M. A. (2018). Going with the flow: A graph based approach to optimal path planning in general flows. *Autonomous Robots*, *42*(7), 1369–1387. <https://doi.org/10.1007/s10514-018-9741-6>

- [10] Lin, K., Wang, H., Chen, B., & Fortino, G. (2022). Hypergraph-based autonomous networks: Adaptive resource management and dynamic resource scheduling. *IEEE Communications Standards Magazine*, 6(3), 16–22.
- [11] Majeed, A., & Rauf, I. (2020). Graph theory: A comprehensive survey about graph theory applications in computer science and social networks. *Inventions*, 5(1), 10.
- [12] Medova, E. A. (1996). Graph-Theoretical Optimization Methods. In P. Cochrane & D. J. T. Heatley (Eds.), *Modelling Future Telecommunications Systems* (pp. 103–123). Springer US. https://doi.org/10.1007/978-1-4615-2049-8_7
- [13] Meng, L., Shao, Y., Yuan, L., Lai, L., Cheng, P., Li, X., Yu, W., Zhang, W., Lin, X., & Zhou, J. (2025). A Survey of Distributed Graph Algorithms on Massive Graphs. *ACM Computing Surveys*, 57(2), 1–39. <https://doi.org/10.1145/3694966>
- [14] Nadarajah, S., & Cire, A. A. (2020). Network-Based Approximate Linear Programming for Discrete Optimization. *Operations Research*, 68(6), 1767–1786. <https://doi.org/10.1287/opre.2019.1953>
- [15] Olofsson, M., Andersson, G., & Soder, L. (2002). Linear programming based optimal power flow using second order sensitivities. *IEEE Transactions on Power Systems*, 10(3), 1691–1697.
- [16] Rabe, M., Bilan, Y., Widera, K., & Vasa, L. (2022). Application of the linear programming method in the construction of a mathematical model of optimization distributed energy. *Energies*, 15(5), 1872.
- [17] Royle, G. F., & Godsil, C. (2001). *Algebraic graph theory* (Vol. 207). New York: Springer. http://www.cs.cmu.edu/afs/cs/user/gl_miller/public/Scientific-Computing/F-07/RelatedWork/GodsilRoyle163-179.pdf
- [18] Shafiei Dizaji, F., & Shafiei Dizaji, M. (2022). Novel computational mathematical algorithms for structural optimization using graph-theoretical methods. *Engineering Computations*, 39(6), 2391–2423.
- [19] Shakkottai, S., & Srikant, R. (2008). Network optimization and control. *Foundations and Trends® in Networking*, 2(3), 271–379.
- [20] Sihotang, H. T., Riandari, F., & Sihotang, J. (2024). Graph-based Exploration for Mining and Optimization of Yields (GEMOY Method). *Jurnal Teknik Informatika CIT Medicom*, 16(2), 70–81.
- [21] Stanford Large Network Dataset Collection. (2023). https://snap.stanford.edu/data/?utm_source=chatgpt.com
- [22] Trukhanov, S. (2009). *Novel approaches for solving large-scale optimization problems on graphs* [PhD Thesis]. <https://oaktrust.library.tamu.edu/items/a12b49d5-a07f-4a1b-912d-ef0218bb8787>
- [23] Tugnait, J. K. (2021). Sparse graph learning under Laplacian-related constraints. *IEEE Access*, 9, 151067–151079.
- [24] Wang, Y., & Li, K. (2025). *Large Language Models in Operations Research: Methods, Applications, and Challenges* (No. arXiv:2509.18180). arXiv. <https://doi.org/10.48550/arXiv.2509.18180>
- [25] Wilder, B., Ewing, E., Dilkina, B., & Tambe, M. (2019). End to end learning and optimization on graphs. *Advances in Neural Information Processing Systems*, 32. <https://proceedings.neurips.cc/paper/2019/hash/8bd39eae38511daad6152e84545e504d-Abstract.html>
- [26] Yu, X., Yang, S., Wang, Z., Song, S., Ma, H., Cao, Z., & Zhang, X. (2025). LIGHT: Enhancing Learning Path Recommendation via Knowledge Topology-Aware Sequence Optimization. *Proceedings of the 48th International ACM SIGIR Conference on*

- Research and Development in Information Retrieval*, 306–315. <https://doi.org/10.1145/3726302.373002>
- [27] Zhang, X., Ding, B.-W., Xu, X.-X., Li, J.-Y., Zhan, Z.-H., Qian, P., Fang, W., Lai, K.-K., & Zhang, J. (2022). Graph-based deep decomposition for overlapping large-scale optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53(4), 2374–2386.
- [28] Zhu, Y., Xu, W., Zhang, J., Du, Y., Zhang, J., Liu, Q., Yang, C., & Wu, S. (2022). *A Survey on Graph Structure Learning: Progress and Opportunities* (No. arXiv:2103.03036). arXiv. <https://doi.org/10.48550/arXiv.2103.03036>
- [29] Zhu, Y., Yu, J. J., Zhao, X., Liu, Q., Ye, Y., Chen, W., Zhang, Z., Wei, X., & Liang, Y. (2024). ControlTraj: Controllable Trajectory Generation with Topology-Constrained Diffusion Model. *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 4676–4687. <https://doi.org/10.1145/3637528.3671866>

Conflict of Interest Statement: *The authors declare that there is no conflict of interest regarding the publication of this paper.*

Generative AI Statement: *The author(s) confirm that no Generative AI tools were used in the preparation or writing of this article.*

Publishers Note: *All statements made in this article are the sole responsibility of the authors and do not necessarily reflect the views of their affiliated institutions, the publisher, editors, or reviewers. Any products mentioned or claims made by manufacturers are not guaranteed or endorsed by the publisher.*

Copyright © 2026 **Suman Kumar Giri, Dr. Narmata Kaushal**. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author and the copyright owner are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

This is an open access article under the CC-BY license. Know more on licensing on <https://creativecommons.org/licenses/by/4.0/>



Cite this Article

Suman Kumar Giri, Dr. Narmata Kaushal. Integrating Linear Programming and Graph-Theoretic Models for Enhanced Optimization: A Mathematical and Computational Analysis. *International Research Journal of Engineering & Applied Sciences (IRJEAS)*. 14(1), pp. 115-129, 2026. <https://doi.org/10.55083/irjeas.2026.v14i01010>