

Leveraging Machine Learning to Solve Combinatorial **Optimization Problems**

Ahmed Guetti, Osama Baazzi University Toulouse III Paul Sabatier



Efficient CPU-GPU communication

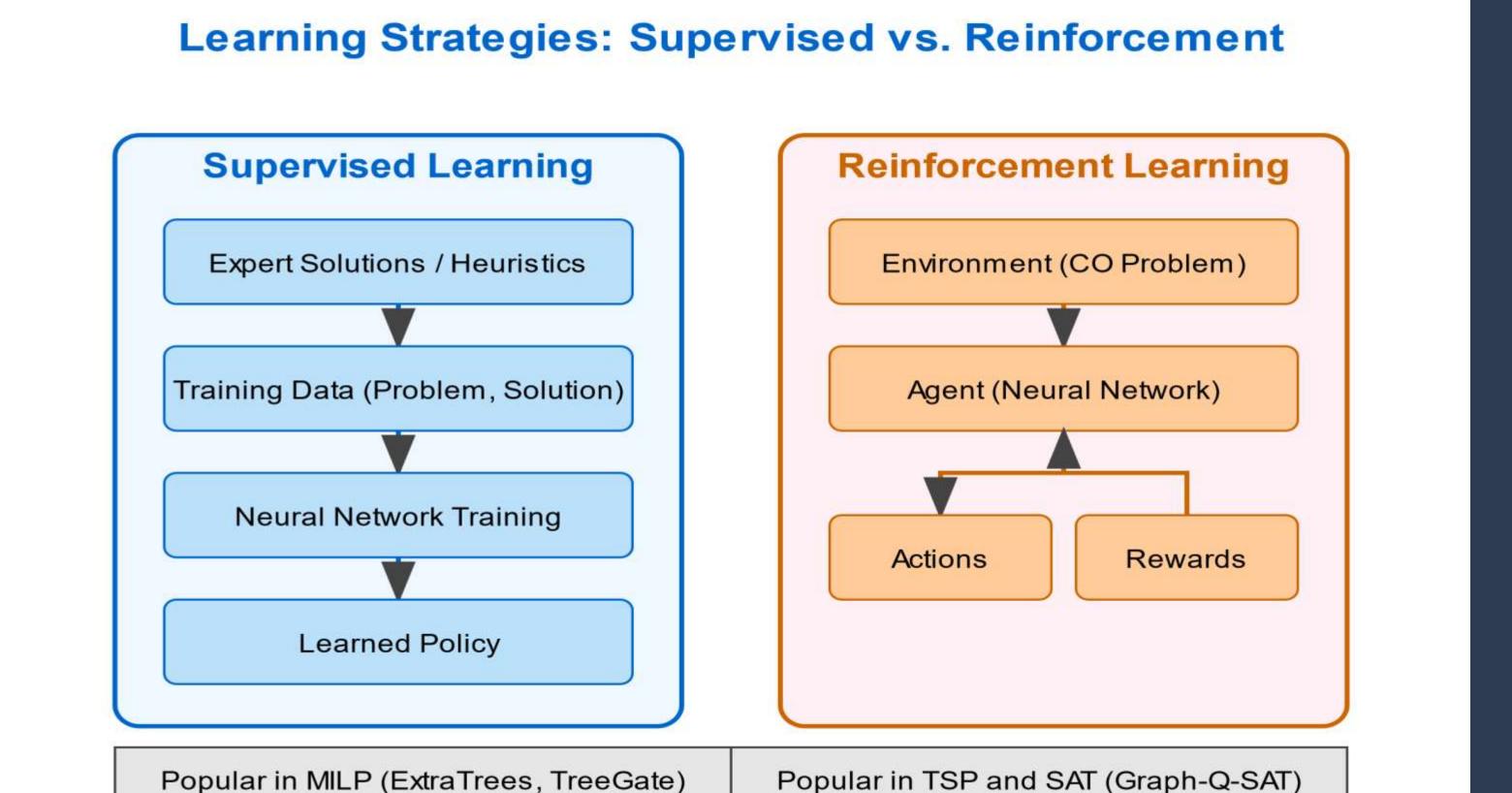
Integration with commercial solvers

Abstract

Learning Strategies

This research reviews recent literature on leveraging machine learning to solve combinatorial optimization problems. We examine three main domains: general combinatorial optimization, Branch-and-Bound methods for Mixed Integer Linear Programming, and Boolean Satisfiability solvers. Our comparative analysis highlights the strengths and limitations of different architectural paradigms and learning strategies.

Introduction **ML Approaches for Combinatorial Optimization Problem Domains** General CO (TSP) MILP (Branch-and-Bound) SAT Solvers **Neural Network Architectures Graph Neural Networks** Hybrid Approaches **Pointer Networks** Neural + Traditional Attention mechanism Message passing Best performance Generates permutations Graph structure ExtraTrees, TreeGate Good for TSP Versatile across domains **Learning Strategies** Reinforcement Learning Supervised Learning Learning through experience Imitation of expert solutions $\pi^* = \arg \max_{\pi} E_{\pi}[\sum \gamma^t R(s_t, a_t)]$ min $E[\ell(Y, f_{\theta}(X))]$ Common in TSP and SAT solvers Common in MILP approaches



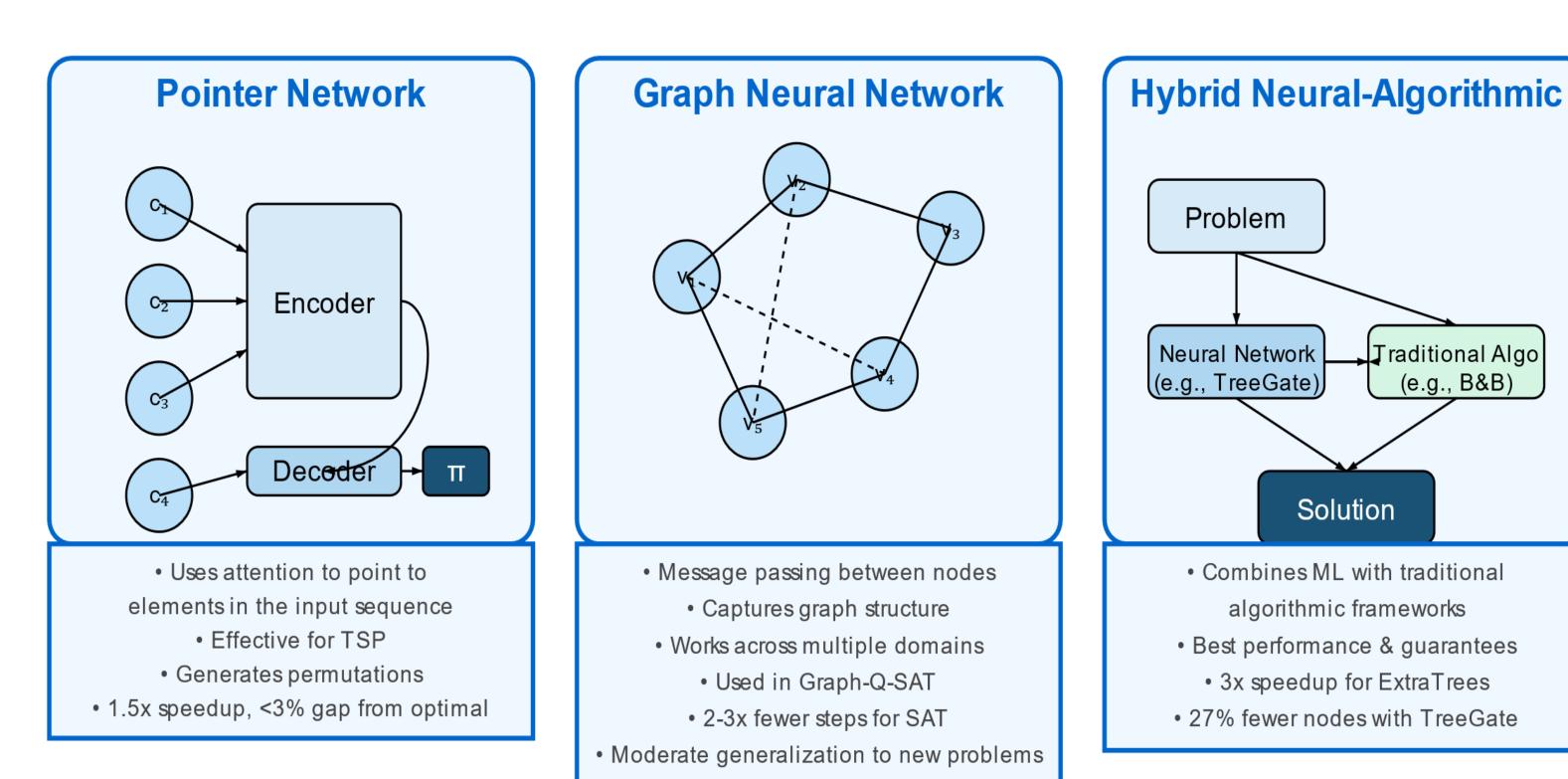
Background & Methods Domain-Specific Findings: ML for Combinatorial Optimization MILP (B&B) **SAT Solvers** TSP CNF Formula. $(X_1 \lor X_2 \lor \neg X_3)$ $(\neg X_1 \lor X_3 \lor X_4)$ ML Guides Branching $(X_2 \lor \neg X_3 \lor \neg X_4)$ **Key Findings Key Findings Key Findings** Pointer Networks show 1.5x speedup ExtraTrees approach achieves 3x Graph-Q-SAT reduces solving GNNs achieve < 2% gap from optimal speedup vs. strong branching iterations by 2-3x Reinforcement learning performs TreeGate shows 27% reduction GNNs effectively capture variable better than supervised for TSP relationships in SAT instances in nodes explored Challenges remain for larger Tree state parameterization RL-based approaches generalize problem instances well across SAT domains improves generalization

Common Pattern: Graph Neural Networks show promise across all domains

Hybrid approaches combining ML with traditional algorithms yield best performance

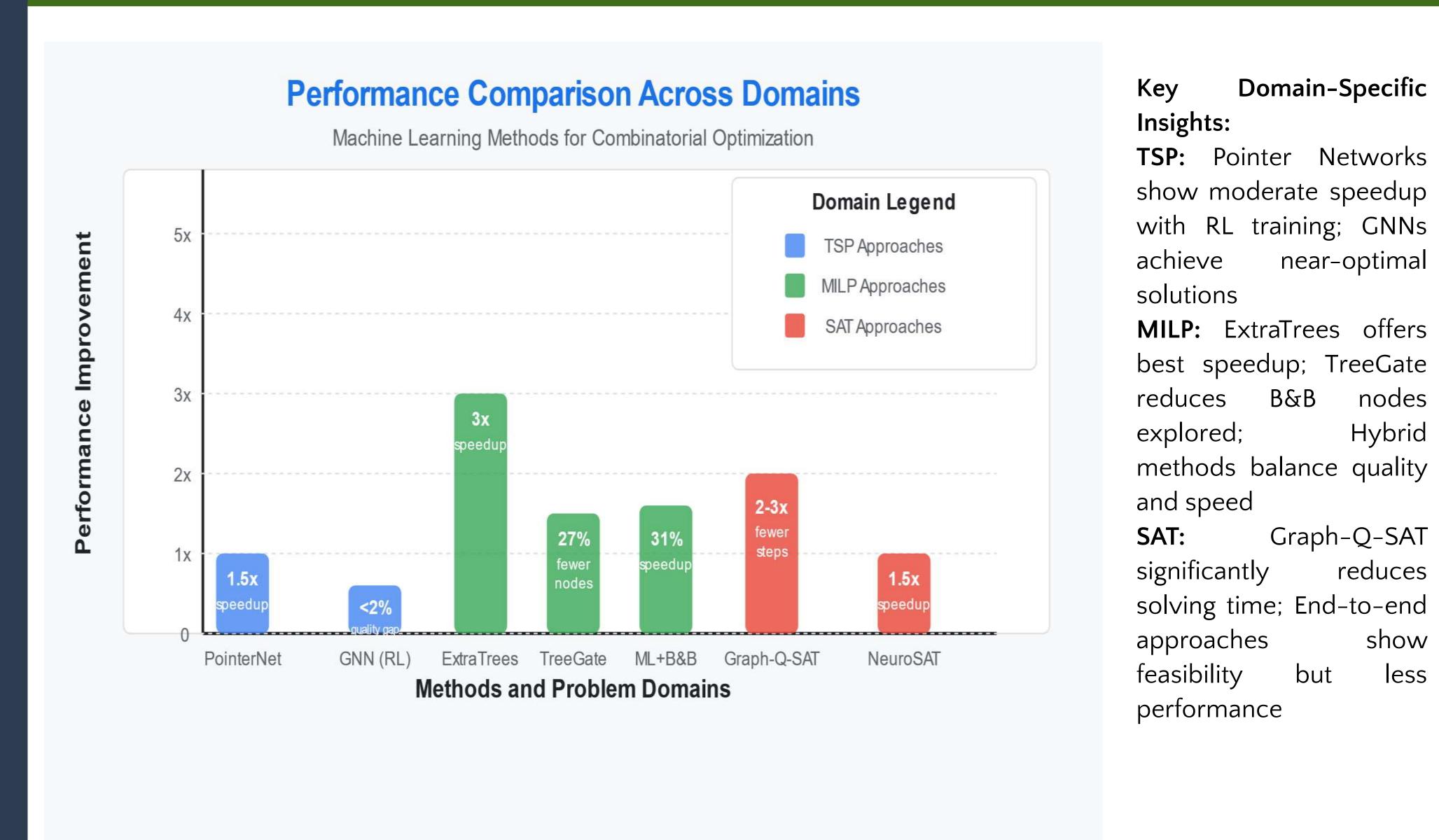
Neural Network Architectures

Neural Network Architectures for Combinatorial Optimization



Hybrid neural-algorithmic approaches show the most promise, Balancing solution quality, computational efficiency, and generalization

Performance Comparison



Note: Performance metrics vary by domain (speedup, fewer nodes, fewer steps)

but

Domain-Specific

Networks

nodes

Hybrid

show

less

near-optimal

Graph-Q-SAT

Insights:

approaches

feasibility

performance

Pointer

Future Research Directions

Future Research Directions in Neural Combinatorial Optimization Neural Specialized GNN architectures Combining SL and RL approaches Combinatoria Attention mechanisms for CO Curriculum learning for CO Optimization Memory-efficient designs Self-supervised pretraining

- Cross-domain knowledge transfer
- · Few-shot learning for new problems
- Size-agnostic model training reduces
- Real-time decision support systems Diverse problem instance libraries Standardized evaluation metrics
 - Cross-architecture comparison tools

Conclusion

Machine learning approaches show promising results for combinatorial optimization, with The use of Graph Neural Networks, hybrid approaches combining ML with traditional algorithms offering the best balance. While significant advances have been made, challenges remain.