Learning to Guide Heuristic Search in Combinatorial Optimization

Günther R. Raidl

Algorithms and Complexity, TU Wien, Austria,
raidl@ac.tuwien.ac.at

University of Banja Luka, Banja Luka, Bosnia and Herzegovina
September 6, 2024

Algorithms and Complexity @ TU Wien

Part of Informatics Faculty @ TU Wien

5 Professors + ≈6 PostDoc + ≈25 PreDoc researchers

Main research areas:

- algorithm design & analysis
- combinatorial optimization
- complexity theory
- computational geometry
- constraint programming
- fixed-parameter algorithms
- graph algorithms
- graph drawing
- heuristic problem solving
- machine learning
- mathematical programming
- SAT solving

Main Research Interests of G. R.

- Combinatorial optimization
- Metaheuristics including evolutionary methods
- Mathematical programming
  - incl. mixed-integer linear programming, column generation, branch-and-cut-and-price, (logic-)based Benders decomposition
- Constraint programming
- Machine learning
- **Hybrid approaches** incl. matheuristics, learning + classical algorithms for COP

Application areas:

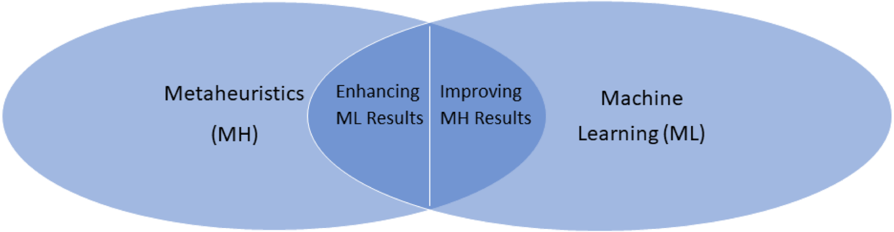
- Transport optimization
- Scheduling
- Network design
- Problems in bioinformatics
- Cutting and packing

Selected Ongoing Projects

- Solving Roman Domination Problems, Influence Maximization Problems, and Variants
  - with M. Djukanovic et al., Univ. of Banja Luka, Bosnia and Herzegovina
- Dynamic Vehicle Routing Problems with Focus on E-mobility & Learning
  - with T. Rodemann et al., Honda Research Institute Europe
- Cooperative Personnel Scheduling
  - with S. Limmer et al., Honda Research Institute Europe
- Doctoral College Vienna Graduate School on Computational Optimization
  - with University of Vienna, IST Austria, Vienna University of Economics and Business
- Catalyst: International Leaders Fellowship Grant
  - with Royal Society of New Zealand, Research Trust of Victoria University of Wellington

Combinatorial Optimization and Learning

- AI/machine learning boom also hit the area of combinatorial optimization
- This in many different ways



- Focus here: utilize learning to better solve combinatorial optimization problems (COPs)

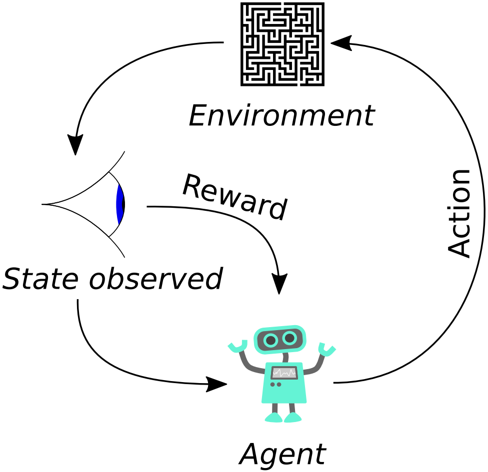
Some Classical Metaheuristics Involving Learning

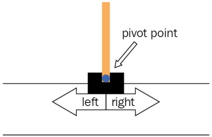
Basic idea of learning in (meta-)heuristics not new:

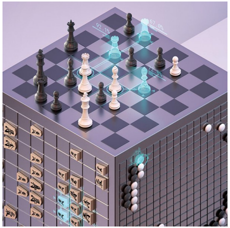
- Reactive tabu search
- Evolution Strategies
- Guided Local Search
- Variable Neighborhood Search,
  Adaptive Large Neighborhood Search
  - self-adaptive selection of neighborhood structures/operators
- Hyper-heuristics
- Ant Colony Optimization

Reinforcement Learning (RL)

- A sub-discipline of machine learning
- Environment is usually considered a Markov decision process
- Framework:

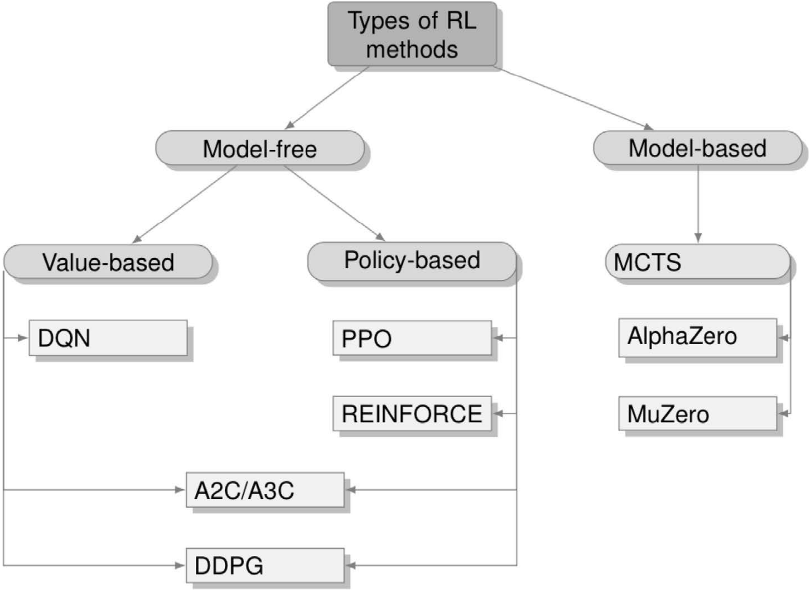






Constructing a solution to a COP can be seen as an episode in an environment, objective value $\hat{=}$ reward

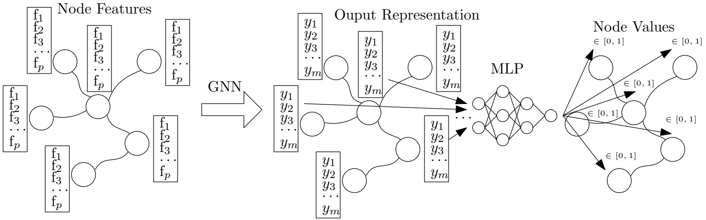
Reinforcement Learning (RL) - Classification



(from Mazyavkina et al. (2021))

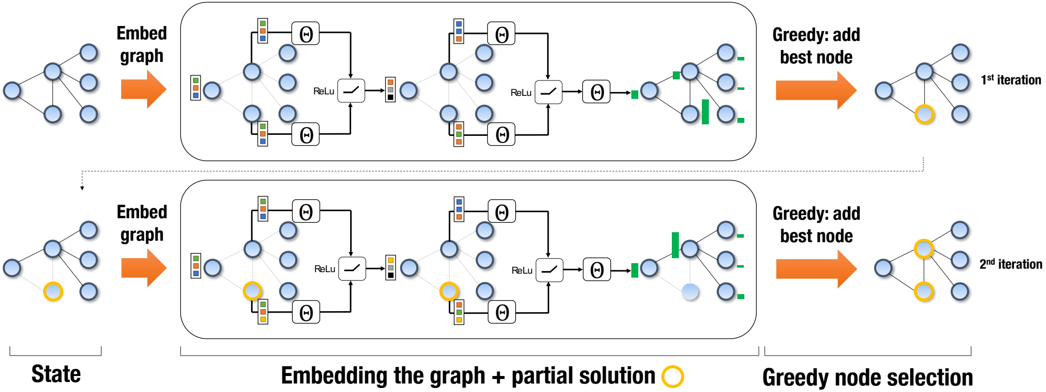
Encoding of Problems+States, ML Models

- encoding highly problem-specific
- variants of (deep) neural networks dominate the used ML models
  - recurrent neural networks, e.g., LSTMs
  - pointer networks (Vinyals et al., 2015)
  - variants of Graph Neural Networks (Scarselli et al., 2008), e.g.,
    - Structure-to-Vector Network (Dai et al., 2016)
    - Graph Convolutional Network (Kipf and Welling, 2017)
    - Graph Isomorphism Network (Xu et al., 2019)
    - Graph Attention Network (Kool et al., 2019; Joshi et al., 2021)



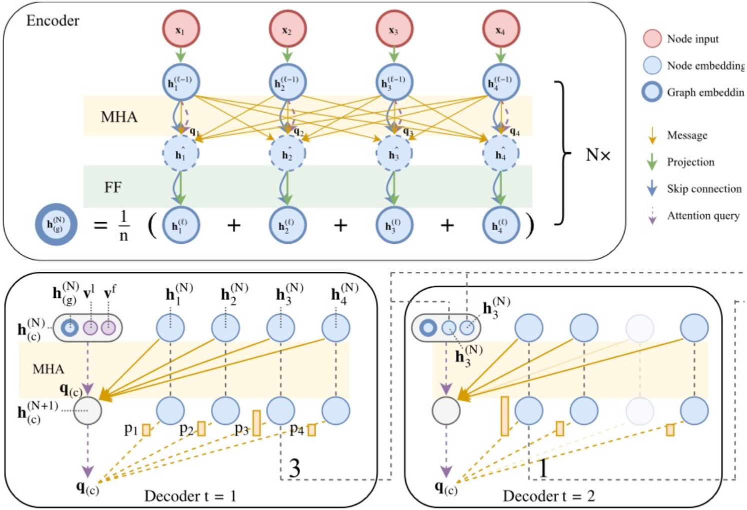
Learning to Solve Graph Problems

- Dai et al. (2017): S2V-DQN
- min vertex cover, max cut, TSP considered
- graph embedding network structure2vec used to “featurize” nodes
- variant of Q-learning used to obtain a policy for greedily constructing solutions



Learning to Solve Graph Problems (cont.)

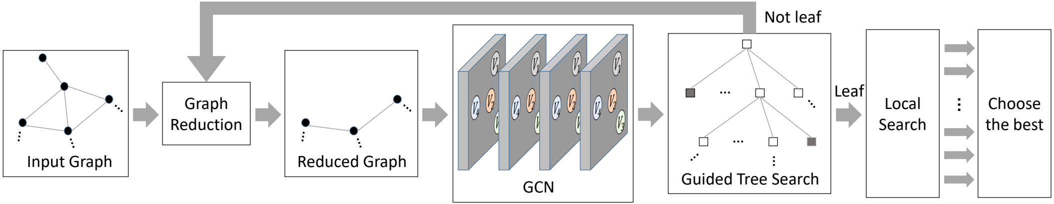
- Kool et al. (2019)
- Autoregressive multi-head attention-based encoder/decoder GNN
- for TSP, VRP



- Trained with REINFORCE

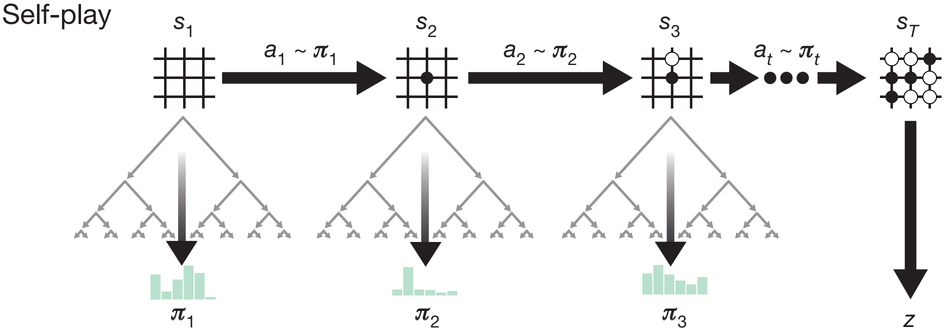
Learning to Solve Graph Problems (cont.)

- Li et al. (2018)
- max independent set, min vertex cover, max clique, SAT considered
- Graph Convolutional Network (GCN) used to predict likelihood of each node to be part of a solution
- GCN yields multiple probability maps to account for the fact that multiple optimal solutions may exist
- heuristic tree search utilizing multiple maps,
  graph reduction, basic local search applied
- supervised learning instead of reinforcement learning
- results competitive to state-of-the-art solvers reported



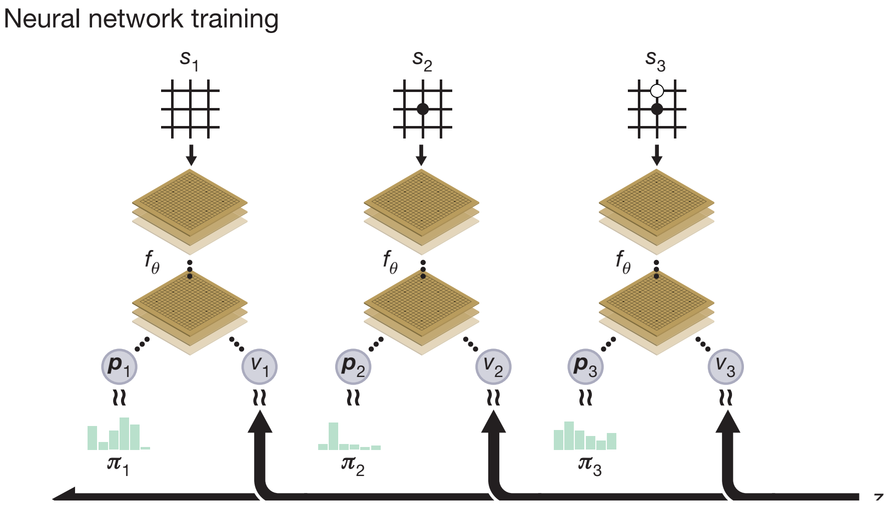
Basic Idea of AlphaGoZero (?)

- Superhuman agent for Go, successor of AlphaGo
- Learns only by iterated selfplay:



- Monte Carlo Tree Search (MCTS) is applied to obtain a policy and select a move
- In the MCTS new states are evaluated by a deep neural net:
  - input: board state
  - output: policy, i.e., probabilities for all positions; value, i.e., probability to win
- Neural net output is **boosted** by MCTS!

Basic Idea of AlphaZero (Silver et al., 2018)



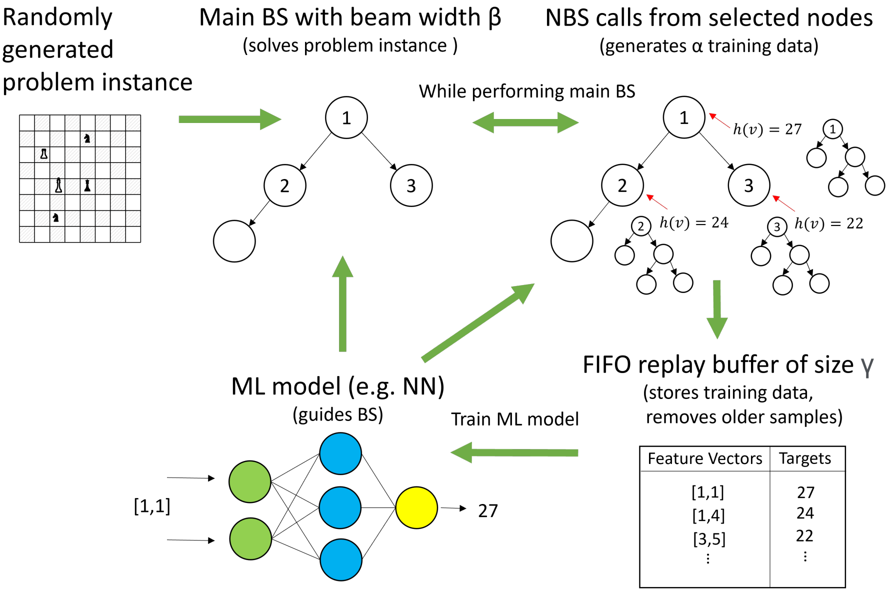
- selfplay games are logged with results in a replay buffer
- neural net continuously trained with samples from replay buffer

Learning to Solve Graph Problems

- **Abe et al. (2020)**: CombOptZero
- min vertex cover, max cut, max clique problems considered
- based on the principles of AlphaGoZero
- different graph neural networks tested, including GCN
- special reward normalization applied
- outperforms S2V-DQN, results close to state-of-the-art reported

- **Huang et al. (2019)**: similar approach for coloring large graphs with millions of nodes
- special FastColorNet neural network architecture
- claimed to yield new state-of-the-art results

Learning Beam Search (Huber and Raidl, 2021)



Feature Vectors	Targets
[1,1]	27
[1,4]	24
[3,5]	22
⋮	⋮

Longest Common Subsequence Problem

Given: set of m input strings $S = \{s_1, \ldots, s_m\}$ over alphabet Σ.

- Longest Common Subsequence (LCS): find a longest string that appears as subsequence in any string of S.

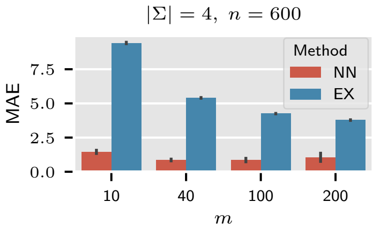
  Example: $m = 2$, $|\Sigma| = 3$

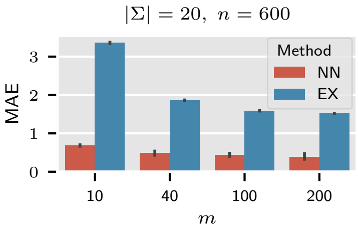
  s_1: ABBA
  s_2: CABA
  $\Rightarrow$ ABA.

State-of-the-art: BS with theoretically derived guidance functions EX (Djukanovic et al., 2020)

LBS Experiments: Approximation of Real LCS Length

The learned network of LBS approximates the real expected LCS lengths better than EX:





LBS Experiments: Results

Results on `rat` and `BB` LCS benchmark instances:

- NN: MLP with 20+20 hidden nodes
- Features: remaining input string lengths, remaining min. letter occurrences
- Beam width:
  - LBS training done with $\beta = 50$
  - Low computation time tests with $\beta = 50$
  - High quality tests with $\beta = 600$

LBS achieved new best results in

- low time experiments: 13 out of 28
- high quality experiments: 7 out of 28

and matched most others.

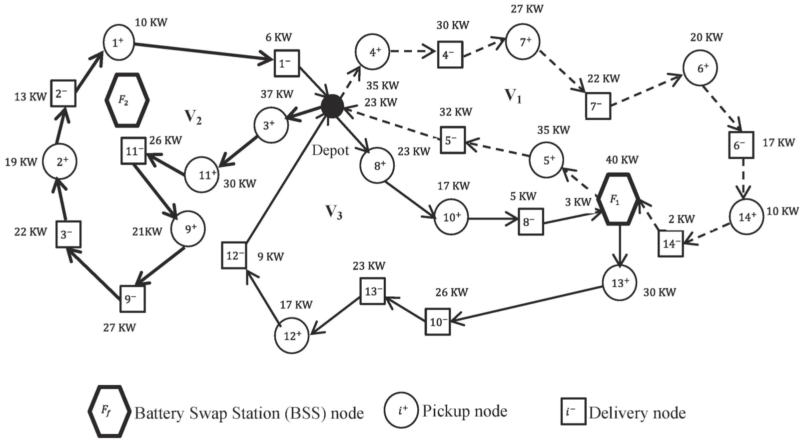
Also successfully considered:
Constrained LCS, shortest common supersequence problem, no-wait flow shop problem

The Electric Autonomous Dial-a-Ride Problem (EADARP)

(Bongiovanni et al., 2019)

Given: n users with transportation requests from a pickup to a drop-off location, a fleet of m electric autonomous vehicles

Task: Design m vehicle routes serving all requests, s.t. the total travel time and the **excess ride times** of all users are minimized and certain constraints are satisfied.



Large Neighborhood Search for EADARP

(Bresich et al., 2024; GECCO 2024)

- Key-feature: an efficient algorithm to insert charging station visits into routes on-the-fly
- Leading for benchmark instances from literature with up to 100 users, 8 vehicles

Large Neighborhood Search for EADARP

(Bresich et al., 2024; GECCO 2024)

- Key-feature: an efficient algorithm to insert charging station visits into routes on-the-fly
- Leading for benchmark instances from literature with up to 100 users, 8 vehicles

However:

- Limmer (2023): Simpler and faster LNS also applicable to instances with few hundred vehicles, several thousand users
- Our LNS only achieves few iterations within time-limit, gaps 10–30%
- **How to scale up our LNS?**

Sparsening/Clustering Techniques for EADARP

Sparsening to k-nearest neighbor graph or clustering into separate geographical regions:

Does not work at all. – Why?

Sparsening/Clustering Techniques for EADARP

Sparsening to k-nearest neighbor graph or clustering into separate geographical regions:

Does not work at all. – Why?

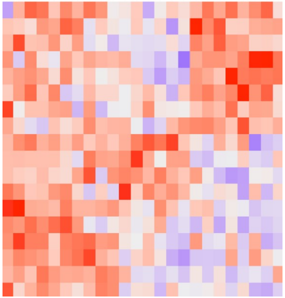
Each order has

- a pickup location
- a dropoff location
- a time window

and orders need to be combined to tours; moreover charging not considered

Learning Heatmaps

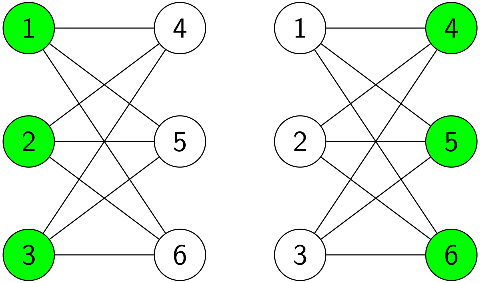
- Learn model indicating likelihood for
  - pairs of orders to be served successively in same tour
  - in (close to) optimal solutions.



- Trained model on medium-sized instances and solutions obtained by the LNS
- Diverse classical ML models as well as small neural networks considered; reasonable results obtained
- More substantial improvements achieved with graph neural networks

Potential Issue of Heatmaps: Unimodality

Example: Maximum independent set problem on $K_{3,3}$ has two optimal solutions:



Heatmap: all nodes are equally likely in an optimal solution.

$\rightarrow$ no meaningful information

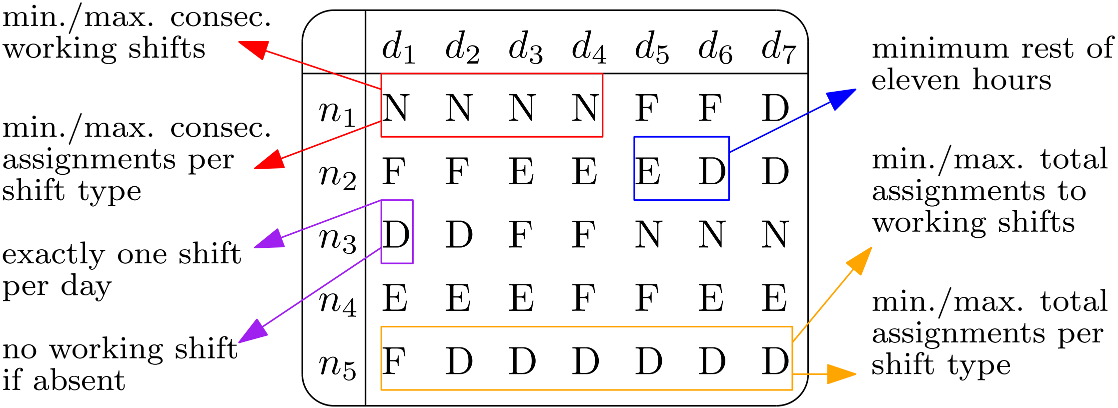
More generally, symmetries and very different (close to) optimal solutions may cause problems.

Learning Effective Destroy Sets in LNS

- Decomposition-based learning LNS
  (Song et al., 2020)
- Neural LNS
  (Addanki et al., 2020)
- Neural Neighborhood Selection (NNS)
  (Sonnerat et al., 2021)
- Learning Large Neighborhood Search for Staff Rerostering
  (Oberweger et al., 2022)

Staff Rerostering Problem (SRRP)

- **Given:** old schedule, disruptions, demand to be met
- **Goal:** create new schedule
  - meeting new demand as best as possible (soft)
  - having as few changes to old schedule as possible (soft)
  - meeting all hard constraints, e.g., work regulations

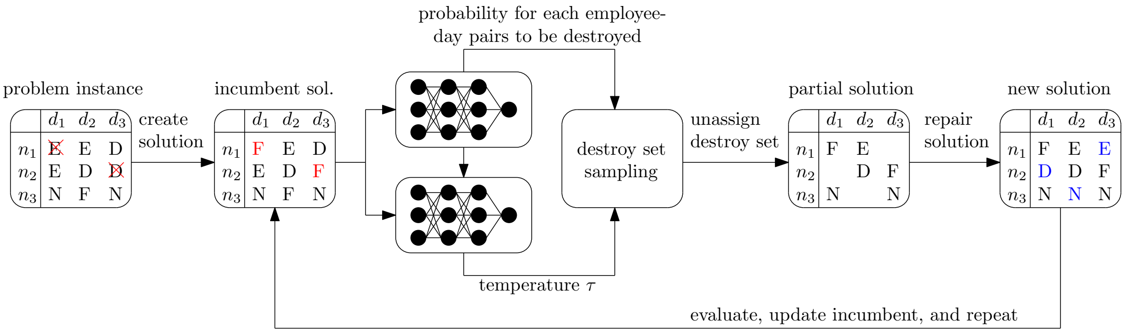


	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	N	N	N	N	F	F	D
n_2	F	F	E	E	E	D	D
n_3	D	D	F	F	N	N	N
n_4	E	E	E	F	F	E	E
n_5	F	D	D	D	D	D	D

Figure: Overview of hard constraints.

Learning LNS for SRRP

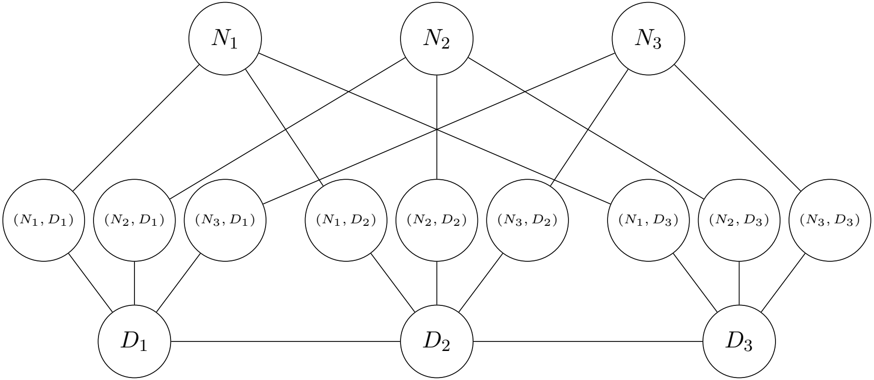
- Initial solution from a simple construction heuristic
- Destroy: Unassign some variables → partial solution



- Repair: Mixed Integer Linear Programming (MILP) solver applied
- Training: Supervised, optimal destroy sets from MILP model with local branching constraint

Learning-Based Destroy Operator

- Model current solution as a graph in each state of LNS
- Use Graph Neural Network (GNN)
- Predict probability of each employee-day pair to belong to destroy set yielding highest improvement
- Select with randomized sampling procedure enforcing selection of segments



Learning-Based Destroy Operator

Training

- Offline with representative problem instances via imitation learning
- Expert policy:
  MILP with local branching constraint to determine optimal destroy set
  (very slow)
- Loss function: log-likelihood of expert actions, cross-entropy for temperature
- DAGGER (Ross et al., 2011):
  Trajectories are first created with expert strategy,
  later with learned model

Computational Results

- Model trained with $|N| = 110$ employees
- MILP + Gurobi optimality gap between **26%** and **34%**

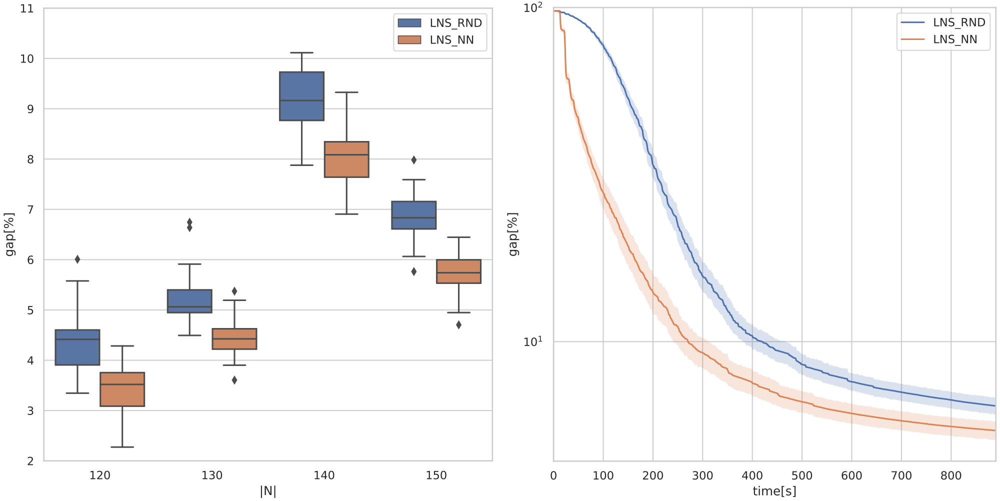


Figure: Comparison of LNS_RND and LNS_NN optimality gaps. 15 minutes running time. Lower bounds from solving MILP for three hours.

Potential Issue of Learning-Based Destroy

- Multimodality:
  Often there are multiple (close to) optimal destroy sets.

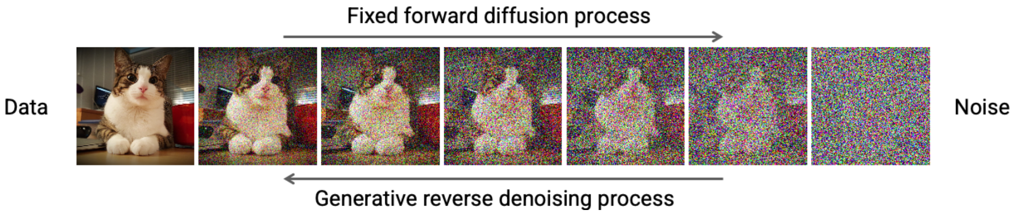
- Learning just with single best destroy set per training sample can be misleading.

Potential Issue of Learning-Based Destroy

- Multimodality:
  Often there are multiple (close to) optimal destroy sets.
- Learning just with single best destroy set per training sample can be misleading.
- Aggregating multiple (close to) optimal destroy sets can be beneficial.
  However: Obtained probability distributions often less informative
- Carefully designed problem-specific sampling procedure important!

Denoising Diffusion Models (DDMs)

- State-of-the-art in many generative AI applications, in particular the creation of realistically-looking images

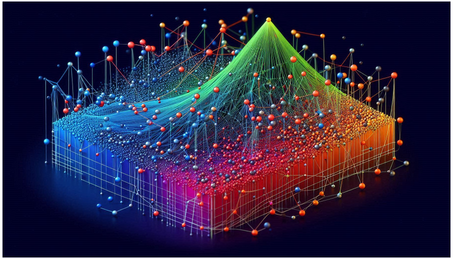


- Training
  - Gaussian noise step-wise added to original images
  - Neural network trained to predict noise added in each step
- Inference
  - Starts from pure random noise
  - Stepwise remove noise via neural network
- DDMs can be conditioned on additional input
- Concept can also be applied to graph neural networks!

DIFUSCO: Graph-Based Diffusion Solver for Combinatorial Opt.

(Sun and Yang, 2023)

- TSP and maximum independent set problem considered
- utilizes an anisotropic graph neural network with edge gating
- discrete diffusion based on Bernoulli noise
- trained on many small instances + (close to) optimal solutions
- used to create diverse heatmaps
- greedy heuristics and MCTS used as decoder



DIFUSCO: Graph-Based Diffusion Solver for Combinatorial Opt.

(Sun and Yang, 2023)

- TSP and maximum independent set problem considered
- utilizes an anisotropic graph neural network with edge gating
- discrete diffusion based on Bernoulli noise
- trained on many small instances + (close to) optimal solutions
- used to create diverse heatmaps
- greedy heuristics and MCTS used as decoder

- Advantages
  - outperforms earlier approaches by a large margin in their tests
  - faster than autoregressive models
  - better scaling behavior to larger instances
  - multi-modality of solution space is considered

DIFUSCO: Graph-Based Diffusion Solver for Combinatorial Opt.

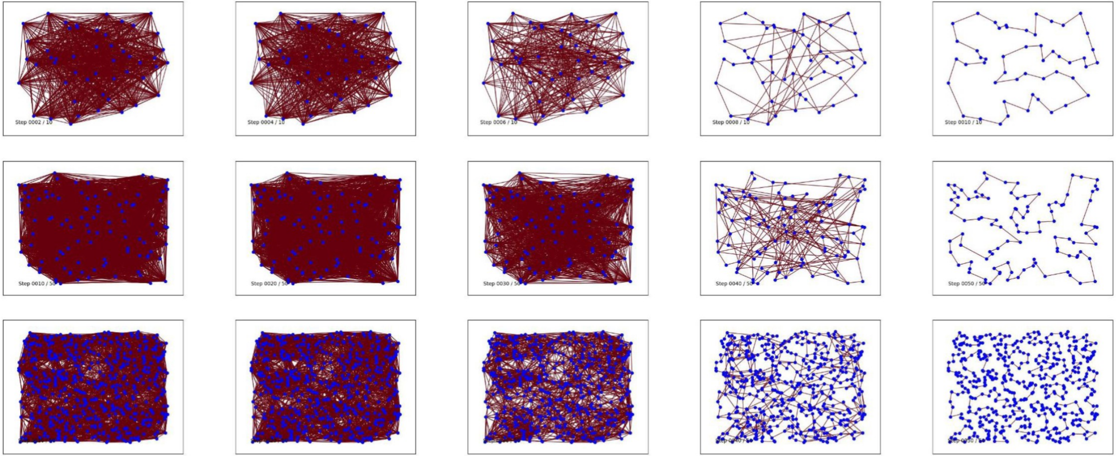


Figure 11: Qualitative illustration of discrete DIFUSCO on TSP-50, TSP-100 and TSP-500 with 50 diffusion steps and `cosine` schedule.

(from Sun and Yang (2023))

DIFUSCO: Graph-Based Diffusion Solver for Combinatorial Opt.

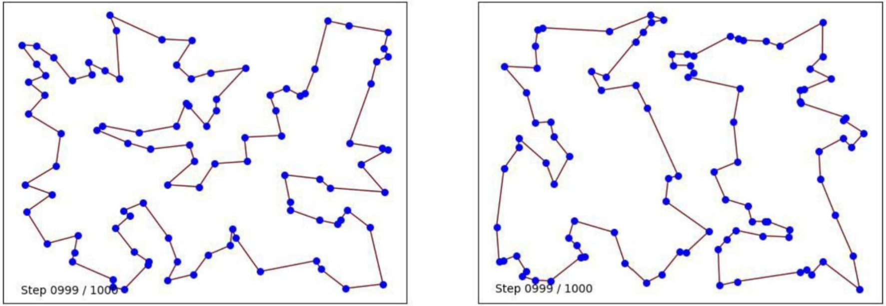


Figure 12: Success (left) and failure (right) examples on TSP-100, where the latter fails to form a single tour that visits each node exactly once. The results are reported without any post-processing.

(from Sun and Yang (2023))

Our Ongoing Work

We are currently investigating DDM & GNN-based approaches for EADARP

- to determine destroy sets in LNS
- to restrict candidate routes for order insertions
- to restrict candidate positions for order insertions
- to dynamically decompose problem instances

Related DDM & GNN-based methods are also investigated on

- α-domination problem
- maximum influence problems in graphs
- graph burning problem

(Very) early results promising!

Conclusions

- Manifold strategies to improve classical solving approaches for COPs by ML
- End-to-end ML approaches will not soon replace classical CO techniques in general
- ML can help substantially to
  - guide tree search or heuristic search
  - sparsify search spaces
  - find better problem decompositions
  - better focus search operators
- Graph & DDM-based approaches appear particularly promising!(?)

References I

Kenshin Abe, Zijian Xu, Issei Sato, and Masashi Sugiyama. Solving np-hard problems on graphs with extended alphago zero. *arXiv:1905.11623 [cs, stat]*, 2020.

Ravichandra Addanki, Vinod Nair, and Mohammad Alizadeh. Neural large neighborhood search. In *Learning Meets Combinatorial Algorithms at Conference on Neural Information Processing Systems*, 2020.

Claudia Bongiovanni, Mor Kaspi, and Nikolas Geroliminis. The electric autonomous dial-a-ride problem. *Transportation Research Part B: Methodological*, 122:436–456, 2019.

Hanjun Dai, Elias B. Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems 31*, pages 6348–6358, 2017.

Marko Djukanovic, Günther R Raidl, and Christian Blum. A beam search for the longest common subsequence problem guided by a novel approximate expected length calculation. In Giuseppe Nicosia et al., editors, *Proc. of the 5th Int. Conf. on Machine Learning, Optimization and Data Science*, volume 11943 of *LNCS*, pages 154–167. Springer, 2020.

Jiayi Huang, Mostofa Patwary, and Gregory Diamos. Coloring big graphs with AlphaGoZero. *arXiv:1902.10162 [cs]*, 2019.

M. Huber and Günther R. Raidl. Learning beam search: Utilizing machine learning to guide beam search for solving combinatorial optimization problems. In *Machine Learning, Optimization, and Data Science – 7th International Conference, LOD 2021*, volume 11943 of *LNCS*. Springer, 2021. to appear.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! *arXiv:1803.08475 [cs, stat]*, 2019.

Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in Neural Information Processing Systems 31*, pages 539–548. Curran Associates, Inc., 2018.

References II

Steffen Limmer. Bilevel large neighborhood search for the electric autonomous dial-a-ride problem. *Transportation Research Interdisciplinary Perspectives*, 21:100876, 2023.

Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, 2021.

Fabio F. Oberweger, Günther R. Raidl, Elina Rönnberg, and Marc Huber. A learning large neighborhood search for the staff rerostering problem. In Pierre Schaus, editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research – CPAIOR 2022*, volume 13292 of *LNCS*, pages 300–317. Springer, 2022.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters Chess, Shogi, and Go through self-play. *Science*, 362(6419): 1140–1144, 2018.

Jialin Song, Ravi Lanka, Yisong Yue, and Bistra Dilkina. A general large neighborhood search framework for solving integer linear programs. In *Advances in Neural Information Processing Systems*, volume 33, pages 20012–20023. Curran Associates, Inc., 2020.

Nicolas Sonnerat, Pengming Wang, Ira Ktena, Sergey Bartunov, and Vinod Nair. Learning a large neighborhood search algorithm for mixed integer programs. *arXiv preprint arXiv:2107.10201*, 2021.

References III

Zhiqing Sun and Yiming Yang. Difusco: Graph-based diffusion solvers for combinatorial optimization. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 3706–3731. Curran Associates, Inc., 2023.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer Networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 2692–2700. Curran Associates, Inc., 2015.

Features for Learning-Based Destroy Operator

For each assignment (n, d)

- flag indicating whether employee n is assigned to shift $s \in S$ on day d
- flag indicating whether employee n is assigned to shift $s \in S$ on day d in the original roster
- flag indicating whether employee n is absent on shift $s \in S$ on day d
- flag indicating whether the minimum number of consecutive working days constraint is violated for employee n on day d
- flag indicating whether the maximum number of consecutive working days constraint is violated for employee n on day d
- flag indicating whether the minimum number of consecutive assignment constraint is violated for employee n on day d and shift $s \in S$
- flag indicating whether the maximum number of consecutive assignment constraint is violated for employee n on day d and shift $s \in S$

Features for Learning-Based Destroy Operator

For each employee n

- total number of working assignments of employee n
- total number of working assignments of employee n minus minimum number of working days in the planning horizon ($\alpha_{\min}$)
- maximum number of working days in the planning horizon ($\alpha_{\max}$) minus total number of working assignments of employee n
- total number of assignments to shift $s \in S$ of employee n
- total number of assignments to shift $s \in S$ of employee n minus minimum allowed number of assignments to this shift s ($\gamma_s^{\min}$)
- maximum allowed number of assignments to shift $s \in S$ ($\gamma_s^{\max}$) minus total number of assignments to this shift s of employee n
- total number of whole day absences of employee n
- total number of absences per shift $s \in S$ of employee n

For each Day d

- total number of assignments to each shift $s \in S$ on day d
- total number of assignments to each shift $s \in S$ on day d minus cover requirements for this shift s on day d (R^{c}_{ds})