(Reinforcement) Learning for Guiding Metaheuristics

Günther R. Raidl

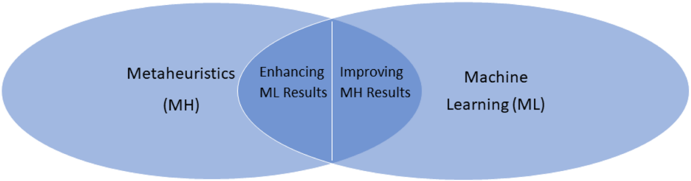
Institute of Logic and Computation, TU Wien, Austria,
raidl@ac.tuwien.ac.at

ELLIIT Hybrid AI Seminar
November 3, 2022

TU WIEN Informatics ac ALGORITHMS AND COMPLEXITY GROUP

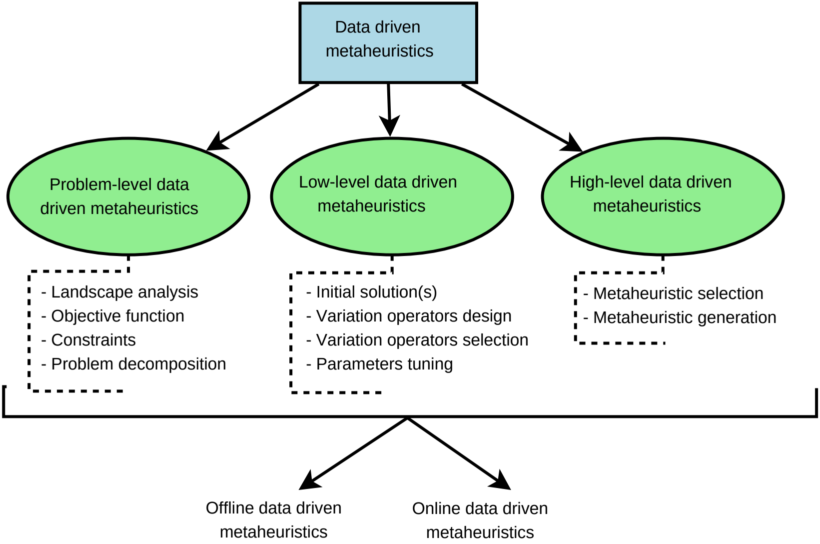
Combinatorial Optimization and Learning

- 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) in heuristic way
- Basic idea of learning in MHs not new!

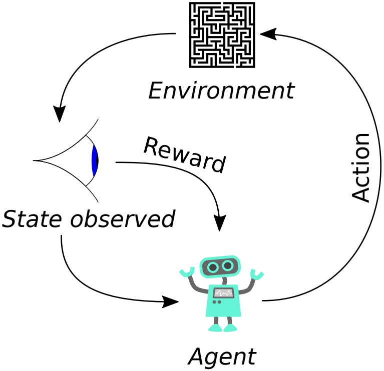
Learning to Better Optimize

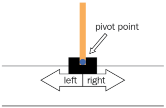


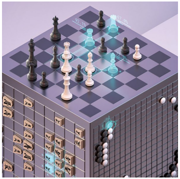
(from Talbi (2021))

Reinforcement Learning (RL)

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

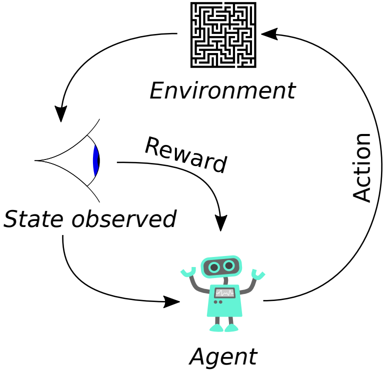


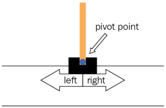


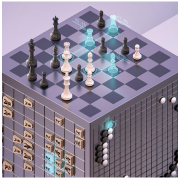


Reinforcement Learning (RL)

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

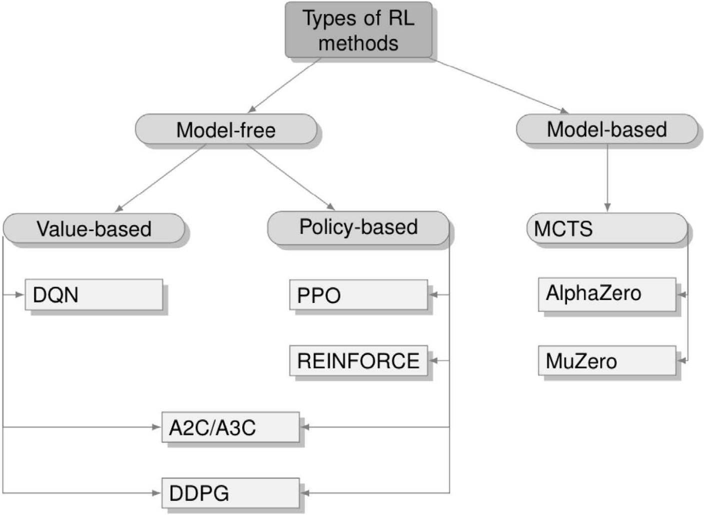






- Constructing a solution to a COP can be seen as an episode in an environment, objective value $\hat{=}$ reward

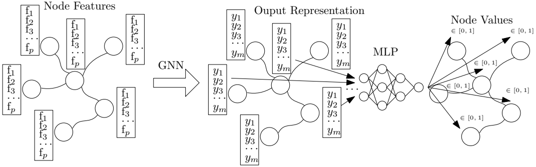
Reinforcement Learning (RL) - Classification



(from Mazyavkina et al. (2021))

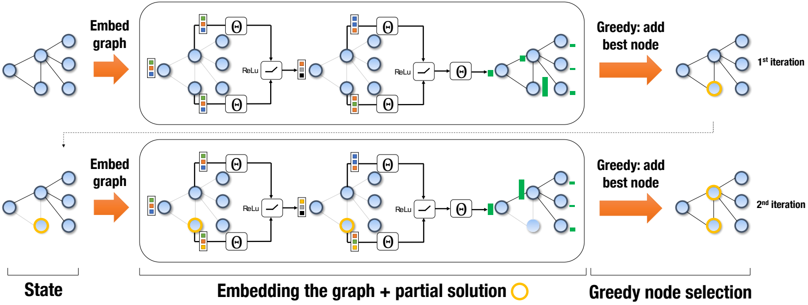
Encoding of Problems+States, ML Models

- 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)



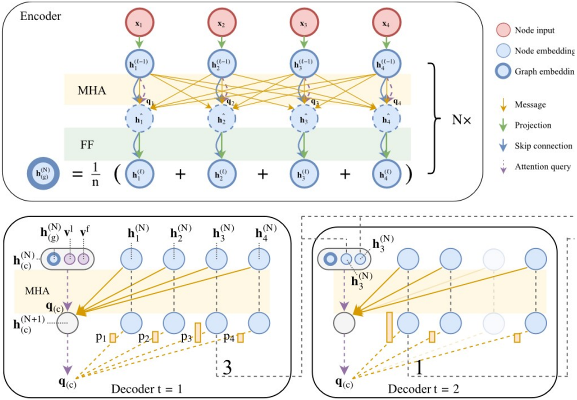
Learning to Solve Graph Problems

- 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.)

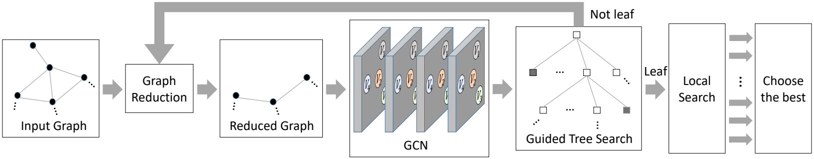
- Kool et al. (2019)
- Autoregressive multi-head attention-based encoder/decoder GNN
- for TSP, VRP



- Trained with REINFORCE

Learning to Solve Graph Problems (cont.)

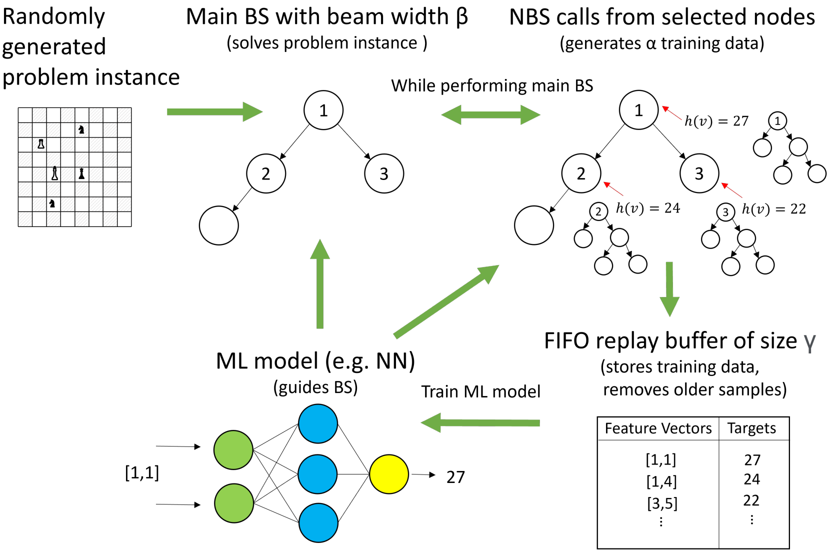
- Li et al. (2018)
- max independent set, min vertex cover, max clique, SAT considered
- Graph Convolutional Network (GCN) used to predict likelihood of each node to be part of a solution
- GCN yields multiple probability maps to account for the fact that multiple optimal solutions may exist
- heuristic 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



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

Learning Beam Search (Huber and Raidl, 2021)



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

A Learning Large Neighborhood Search for the Staff Rerostering Problem

F. Oberweger, G. Raidl, E. Rönnberg, and M. Huber
CPAIOR 22

Related Work

- Large Neighborhood Search (LNS)
  (Pisinger and Ropke, 2010)
- Decomposition-based learning LNS
  (Song et al., 2020)
- Neural LNS
  (Addanki et al., 2020)
- Neural Neighborhood Selection (NNS)
  (Sonnerat et al., 2021)

- Our approach builds on NNS

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

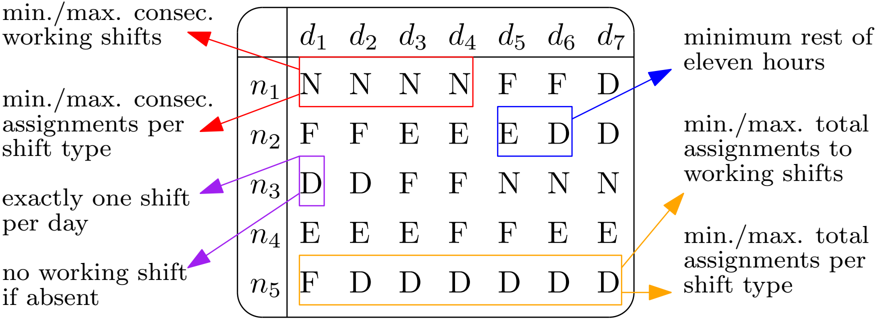
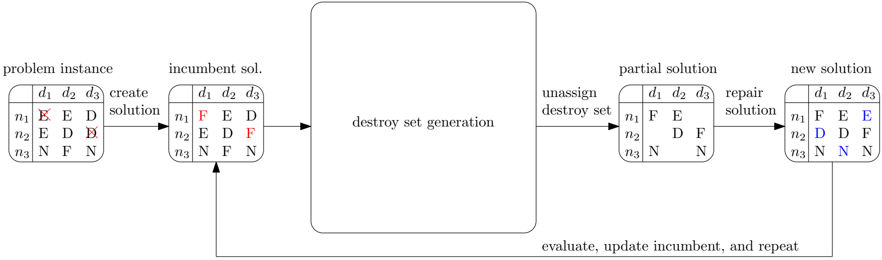


Figure: Overview of hard constraints.

Large Neighborhood Search (LNS)

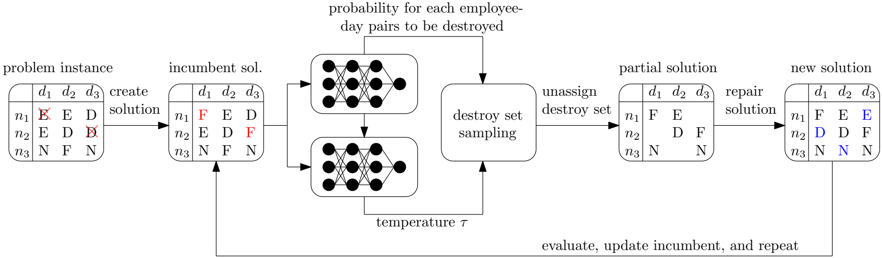
- Initial solution from a simple construction heuristic
- Repeated application of a destroy and a repair operators



- Repair: Mixed Integer Linear Programming (MILP) solver applied

Large Neighborhood Search (LNS)

- Initial solution from a simple construction heuristic
- Repeated application of a destroy and a repair operators



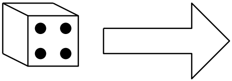
- Repair: Mixed Integer Linear Programming (MILP) solver applied
- Aiming to create a learning-based destroy operator

Repair Operator

- Regular MILP for feasible solutions
- MILP with relaxed hard constraints for infeasible solutions
  - Hard constraint violations are penalized
  - Objective value always worse for infeasible solution

Classical Randomized Destroy Operator

- Randomly choose employee-day pairs
- Destroy all variables associated with employee-day pairs

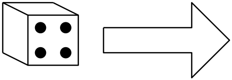


	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	~~N~~	~~N~~	~~N~~	N	F	F	D
n_2	F	F	E	E	E	~~E~~	E
n_3	D	D	F	F	N	N	N
n_4	E	~~E~~	E	F	F	E	E
n_5	F	D	D	D	~~D~~	~~D~~	D

Figure: Destroy operator applied on an example SRRP instance.

Classical Randomized Destroy Operator

- Randomly choose employee-day pairs
- Destroy all variables associated with employee-day pairs

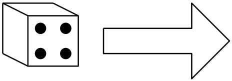


	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	~~N~~	~~N~~		N	F	F	D
n_2	F	F	E	E	E		E
n_3	D	D	F		N	N	N
n_4	E		E	F	F	E	
n_5	F	D	D	D		~~D~~	D

Figure: Destroy operator applied on an example SRRP instance.

Classical Randomized Destroy Operator

- Consecutive day constraints: selecting consec. days unlikely
- Better select and destroy random sequences of days!

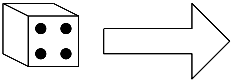


	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	~~N~~	~~N~~	~~N~~	N	F	F	D
n_2	F	F	E	E	E	~~E~~	E
n_3	D	D	F	F	N	N	N
n_4	E	~~E~~	E	F	F	E	E
n_5	F	D	D	D	~~D~~	~~D~~	D

Figure: Destroy operator applied on an example SRRP instance.

Classical Randomized Destroy Operator

- Consecutive day constraints: selecting consec. days unlikely
- Better select and destroy random sequences of days!



	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			
n_3	D	D	F	F	N	N	N
n_4				F	F	E	E
n_5	F	D	D	D	~~D~~	~~D~~	D

Figure: Destroy operator applied on an example SRRP instance.

Learning-Based Destroy Operator

Destroy Set Model

- Use Graph Neural Network (GNN) Scarselli et al. (2008)
- Model current solution as a graph in each state of LNS
- Predict weight of an employee-day pair to belong in destroy set

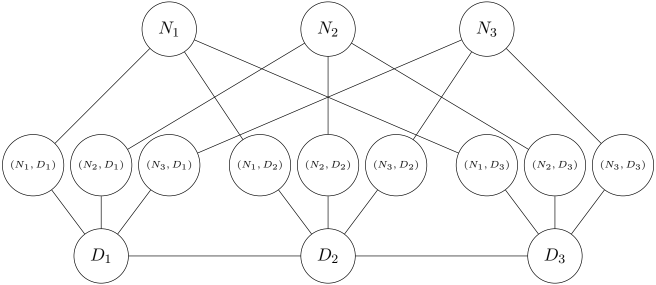


Figure: Simplified representation of the destroy set model architecture.

Learning-Based Destroy Operator

Destroy Set Model

- Use Graph Neural Network (GNN) Scarselli et al. (2008)
- Model current solution as a graph in each state of LNS
- Predict weight of an employee-day pair to belong in destroy set

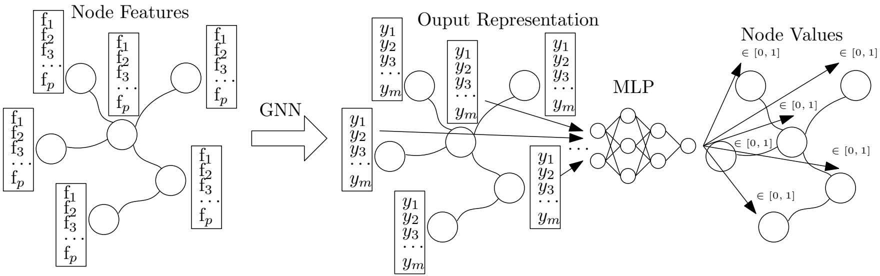


Figure: Simplified representation of the destroy set model architecture.

Features

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 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})

Learning-Based Destroy Operator

Destroy Set Sampling Strategy

- Based on consecutive day observation
- Use GNN outputs $\mu_{nd}\ \forall n \in N, d \in D$ for refined sampling

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	0.1	0.2	0.2	0.5	0.7	0.4	0.2
n_2	0.3	0.1	0.2	0.8	0.1	0.2	0.4
n_3	0.6	0.7	0.2	0.1	0.1	0.5	0.3

$\sum$

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1		0.5					
n_2							
n_3							

Figure: Destroy set sampling strategy.

Learning-Based Destroy Operator

Destroy Set Sampling Strategy

- Based on consecutive day observation
- Use GNN outputs $\mu_{nd}\ \forall n \in N, d \in D$ for refined sampling

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	0.1	0.2	0.2	0.5	0.7	0.4	0.2
n_2	0.3	0.1	0.2	0.8	0.1	0.2	0.4
n_3	0.6	0.7	0.2	0.1	0.1	0.5	0.3

$\sum$

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1		0.5	0.9				
n_2							
n_3							

Figure: Destroy set sampling strategy.

Learning-Based Destroy Operator

Destroy Set Sampling Strategy

- Based on consecutive day observation
- Use GNN outputs $\mu_{nd}\ \forall n \in N, d \in D$ for refined sampling

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	0.1	0.2	0.2	0.5	0.7	0.4	0.2
n_2	0.3	0.1	0.2	0.8	0.1	0.2	0.4
n_3	0.6	0.7	0.2	0.1	0.1	0.5	0.3

$\sum$ →

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	0.3	0.5	0.9				
n_2							
n_3							

Figure: Destroy set sampling strategy.

Learning-Based Destroy Operator

Destroy Set Sampling Strategy

- Based on consecutive day observation
- Use GNN outputs $\mu_{nd}\ \forall n \in N, d \in D$ for refined sampling

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	0.1	0.2	0.2	0.5	0.7	0.4	0.2
n_2	0.3	0.1	0.2	0.8	0.1	0.2	0.4
n_3	0.6	0.7	0.2	0.1	0.1	0.5	0.3

→

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	0.3	0.5	0.9	1.4	1.6	1.3	0.6
n_2	0.4	0.6	1.1	1.1	1.1	0.7	0.6
n_3	1.3	1.5	1.0	0.4	0.7	0.9	0.8

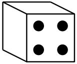
Figure: Destroy set sampling strategy.

Learning-Based Destroy Operator

Destroy Set Sampling Strategy

- Based on consecutive day observation
- Use GNN outputs $\mu_{nd}\ \forall n \in N, d \in D$ for refined sampling

random selection
proportional to weights



	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	0.3	0.5	0.9	1.4	1.6	1.3	0.6
n_2	0.4	0.6	1.1	1.1	1.1	0.7	0.6
n_3	1.3	1.5	1.0	0.4	0.7	0.9	0.8

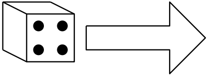
Figure: Destroy set sampling strategy.

Learning-Based Destroy Operator

Destroy Set Sampling Strategy

- Based on consecutive day observation
- Use GNN outputs $\mu_{nd}\ \forall n \in N, d \in D$ for refined sampling

random selection
proportional to weights



	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	0.3	0.5	0.9	1.4	1.6	1.3	0.6
n_2	0.4	0.6	1.1	1.1	1.1	0.7	0.6
n_3	1.3	1.5	1.0	0.4	0.7	0.9	0.8

Figure: Destroy set sampling strategy.

Learning-Based Destroy Operator

Destroy Set Sampling Strategy

- Based on consecutive day observation
- Use GNN outputs $\mu_{nd}\ \forall n \in N, d \in D$ for refined sampling

update underlying weights

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	0.3	0.5	0.4	0.0	0.0	0.0	0.2
n_2	0.4	0.6	1.1	1.1	1.1	0.7	0.6
n_3	1.3	1.5	1.0	0.4	0.7	0.9	0.8

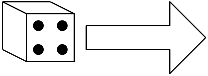
Figure: Destroy set sampling strategy.

Learning-Based Destroy Operator

Destroy Set Sampling Strategy

- Based on consecutive day observation
- Use GNN outputs $\mu_{nd}\ \forall n \in N, d \in D$ for refined sampling

random selection
proportional to weights



	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	0.3	0.5	0.4	0.0	0.0	0.0	0.2
n_2	0.4	0.6	1.1	1.1	1.1	0.7	0.6
n_3	1.3	1.5	1.0	0.4	0.7	0.9	0.8

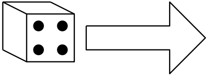
Figure: Destroy set sampling strategy.

Learning-Based Destroy Operator

Destroy Set Sampling Strategy

- Based on consecutive day observation
- Use GNN outputs $\mu_{nd}\ \forall n \in N, d \in D$ for refined sampling

random selection
proportional to weights



	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	0.3	0.5	0.4	0.0	0.0	0.0	0.2
n_2	0.4	0.6	1.1	1.1	1.1	0.7	0.6
n_3	1.3	1.5	1.0	0.4	0.7	0.9	0.8

Figure: Destroy set sampling strategy.

- Regulate influence of GNN with temperature τ
  - Such that $\mu_{nd}^{\frac{1}{\tau}}\ \forall n \in N, d \in D$
  - So far $\tau = 1$

Learning-Based Destroy Operator

Temperature Model

- Learn temperature τ for each state with a GNN
- **Input:**
  - graph representation of current solution
  - destroy set model outputs
- **Output:** probabilities for selecting temperature in $\mathcal{T} = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 2, 3, 5\}$

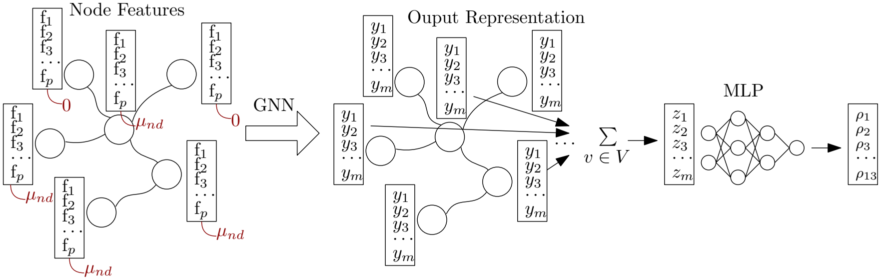


Figure: Simplified representation of the temperature model architecture.

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

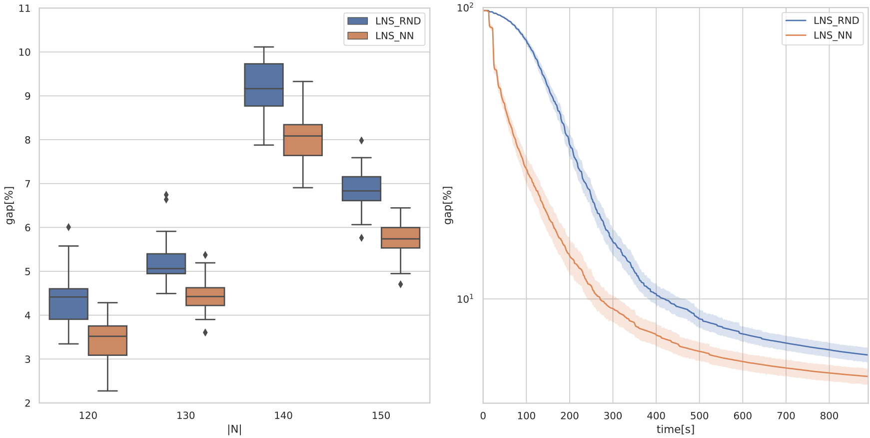


Figure: Comparison of LNS_RND and LNS_NN optimality gaps. 15 minutes running time. Lower bounds from solving MILP for three hours.

Conclusions

- Large variety of ML-based approaches to support/improve metaheuristics
- Modern RL techniques seem particularly promising
  - to reduce effort in manually crafting/tuning heuristics
  - without labeled training data (supervised learning)
- Naive application of an RL agent to a COP usually not competitive
- Combinations with tree search, local search and problem-specific heuristics can boost performance substantially
- Keep in mind:
  - (deep) neural networks not always necessary, e.g., other ML models may be faster & more robust
  - deep RL can be tricky

References I

Abe, K., Xu, Z., Sato, I., and Sugiyama, M. (2020). Solving np-hard problems on graphs with extended alphago zero. *arXiv:1905.11623 [cs, stat]*.

Addanki, R., Nair, V., and Alizadeh, M. (2020). Neural large neighborhood search. In *Learning Meets Combinatorial Algorithms at Conference on Neural Information Processing Systems*.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., and Song, L. (2017). Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems 31*, pages 6348–6358.

Huber, M. and Raidl, G. R. (2021). 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. to appear.

Kool, W., van Hoof, H., and Welling, M. (2019). Attention, learn to solve routing problems! *arXiv:1803.08475 [cs, stat]*.

Li, Z., Chen, Q., and Koltun, V. (2018). Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in Neural Information Processing Systems 31*, pages 539–548.

References II

Mazyavkina, N., Sviridov, S., Ivanov, S., and Burnaev, E. (2021). Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400.

Pisinger, D. and Ropke, S. (2010). Large neighborhood search. In *Handbook of metaheuristics*, pages 399–419. Springer.

Ross, S., Gordon, G., and Bagnell, D. (2011). 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.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Song, J., Lanka, R., Yue, Y., and Dilkina, B. (2020). 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.

Sonnerat, N., Wang, P., Ktena, I., Bartunov, S., and Nair, V. (2021). Learning a large neighborhood search algorithm for mixed integer programs. *arXiv preprint arXiv:2107.10201*.

References III

Talbi, E.-G. (2021). Machine learning into metaheuristics: A survey and taxonomy. *ACM Computing Surveys*, 54(6):129:1–129:32.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer Networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 2692–2700. Curran Associates, Inc.