Evolutionary Computation Meets Machine Learning for Combinatorial Optimisation

Yi Mei[1] and Günther R. Raidl[2]
1 Victoria University of Wellington, New Zealand
2 TU Wien, Vienna, Austria
yi.mei@ecs.vuw.ac.nz
raidl@ac.tuwien.ac.at

http://gecco-2025.sigevo.org/

GECCO '25 Companion, July 14 - 18, 2025, Málaga, Spain

ACM ISBN 979-8-4007-1464-1/25/07.. . $15.00
https://doi.org/10.1145/3712255.3716512

Outline

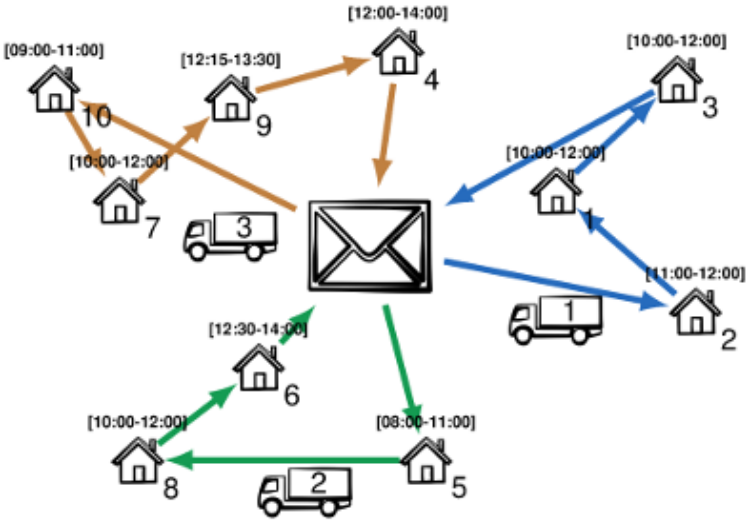
- Introduction and Background
- Evolutionary Computation to Learn Combinatorial Optimisation Heuristics
- Machine Learning to Aid (Meta-)Heuristics
- Challenges and Future Directions

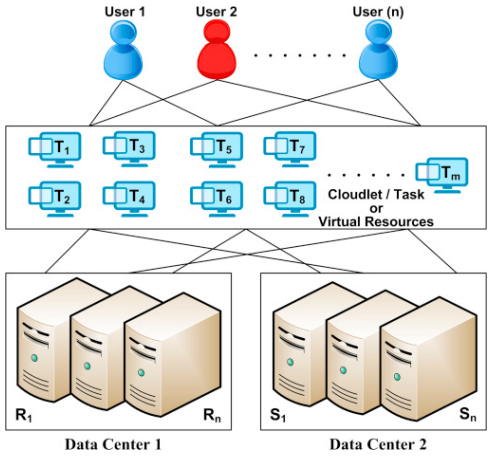
Introduction and Background

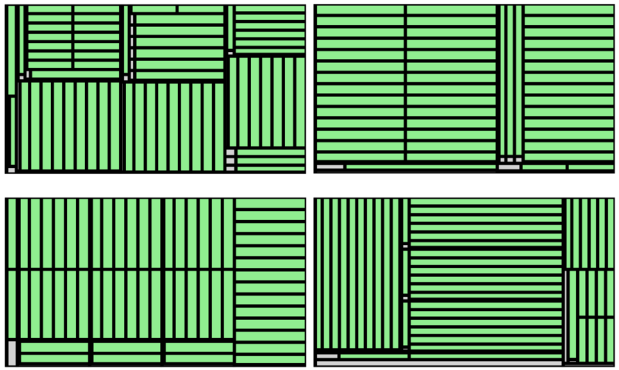
Combinatorial Optimisation

Many real-world applications

- Supply chain/logistics
- Cloud computing
- Manufacturing, scheduling
- Cutting and packing
- Bioinformatics
- …
- *NP-hard, many local optima…*





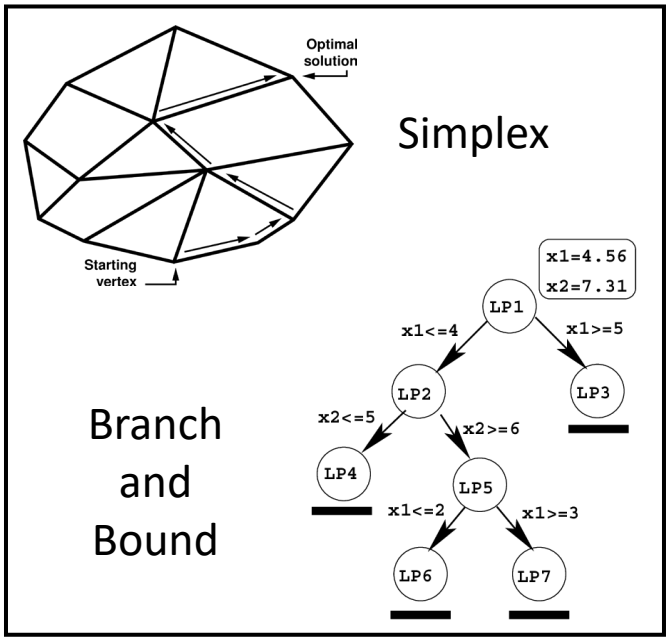


Combinatorial Optimisation Methods

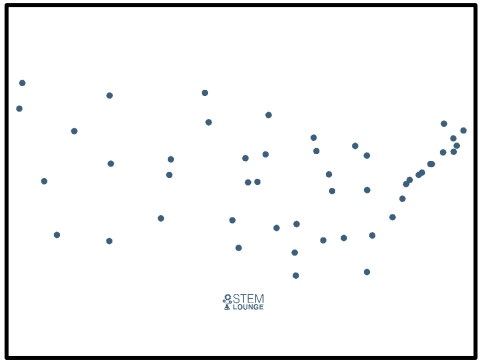
Exact Methods

Constructive heuristics

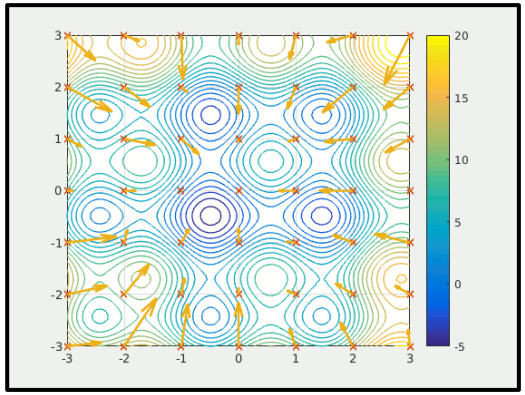
Meta-heuristics



Integer Linear Programming



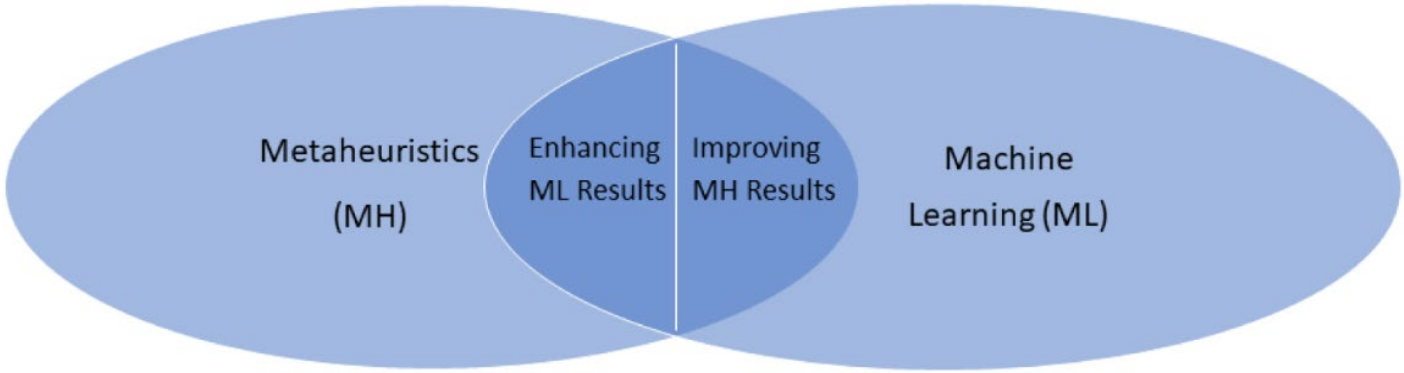
Nearest neighbour heuristic for TSP



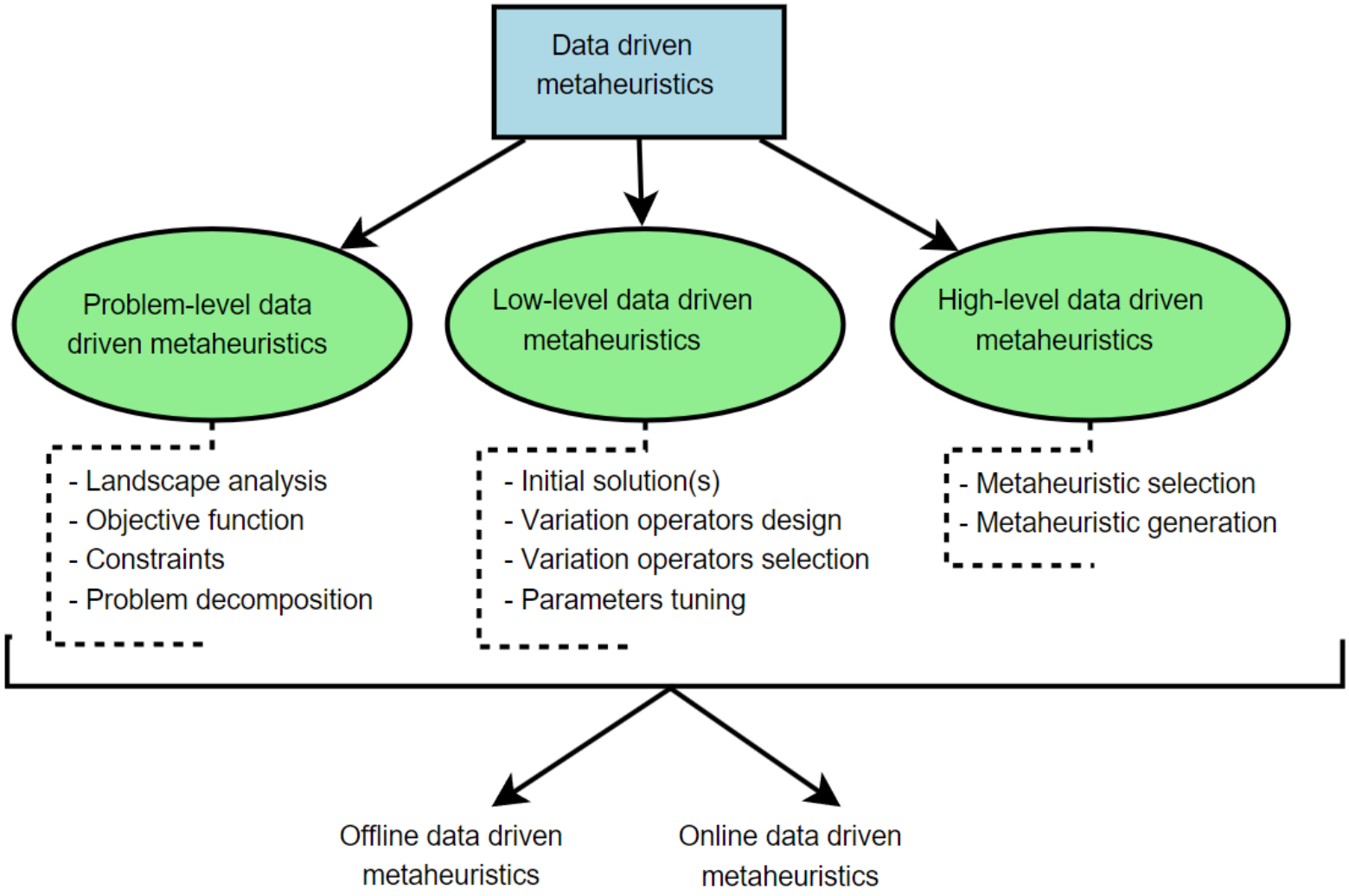
Particle Swarm Optimization

Evolutionary Computation, Metaheuristics, and Learning

- AI/machine learning (ML) boom also hit the area of combinatorial optimization (COP)
- Evolutionary Computation and other (meta-)heuristics relate to machine learning in many different ways



Learning to (Better) Optimize



From Talbi (2021): Machine Learning into Metaheuristics: A Survey and Taxonomy

Survey Articles

- M. Karimi-Mamaghan, M. Mohammadi, P. Meyer, A. M. Karimi-Mamaghan, E.-G. Talbi (2021): Machine Learning at the Service of Metaheuristics for solving COPs: A State-of-the Art, EJOR
- E. G.Talbi (2021): Machine Learning into Metaheuristics: A Survey and Taxonomy, ACM Computing Surveys
- Y. Bengio, A. Lodi, A. Prouvost (2021): Machine Learning for Combinatorial Optimization: A Methodological Tour d'Horizon; EJOR
- N. Mazyavkina, S. Sviridov, S. Ivanov, E. Burnaev (2021): Reinforcement Learning for Combinatorial Optimization: A Survey; COR
- C. Zang, Y. Wu, Y. Ma, W. Song, Z. Le (2023): A Review on Learning to Solve Combinatorial Optimization Problems in Manufacturing, IET Collaborative Intelligent Manufacturing

Some Classical Metaheuristics Involving Learning

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

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

Q-Learning (Watkins and Dayan, 1992)

- for discrete state and action spaces, off-policy RL
- Goal: max. weighted sum of expected rewards over all future steps
- Table with Q-value $Q(s,a)$ for each state/action pair (s,a): approximates expected weighted sum of rewards to receive from state s onward when performing action a
- Update of initially random $Q(s,a)$ values by

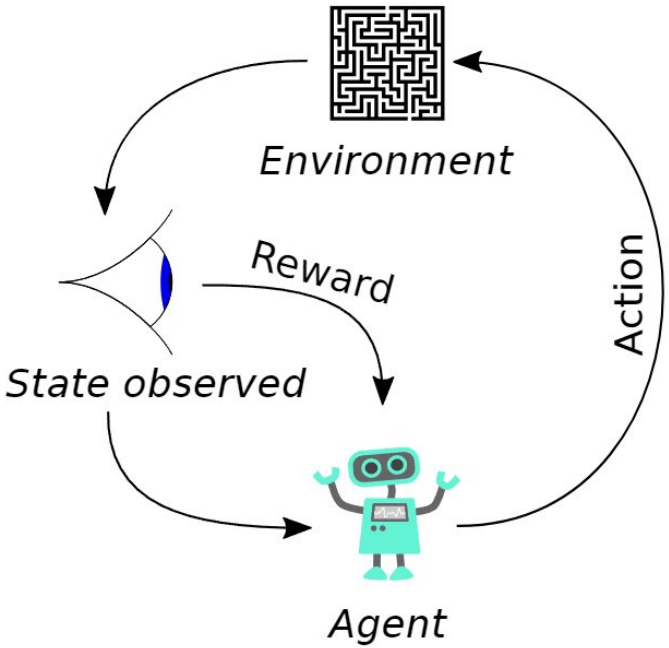
$$Q^{new}(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t,a_t)}_{\text{old value}}\Big)}^{\text{temporal difference}}$$

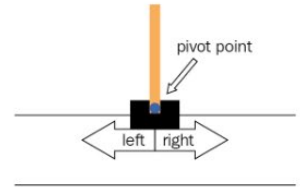
$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{new value (temporal difference target)}}$$

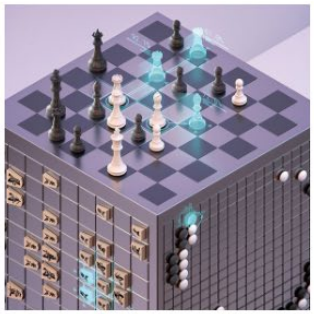
- Compare with **Ant Colony Optimization**!
- Frequently too many states
  - approximate $Q(s,a)$ by a (deep) neural network
  - Deep Q Network (DQN, Mnih et al., 2013)

Reinforcement Learning

- A sub-discipline of machine learning
- Learn to behave well in a given environment
  - usually considered a Markov Decision Process (MDP)
- Framework:







Machine Learning for CO

- Learning to **construct** (initial) solutions
- Learning to **search** in the solution space

- Supervised learning
  - Collect data on problem instances + optimal solutions/steps
  - Learn to mimic the optimal solutions/steps

- Reinforcement learning
  - Given training problem instances, no solution needed
  - Learn a policy to make decisions
  - Reward/Quality indicators
    - How good the policy is
    - Based on solutions obtained by the policy

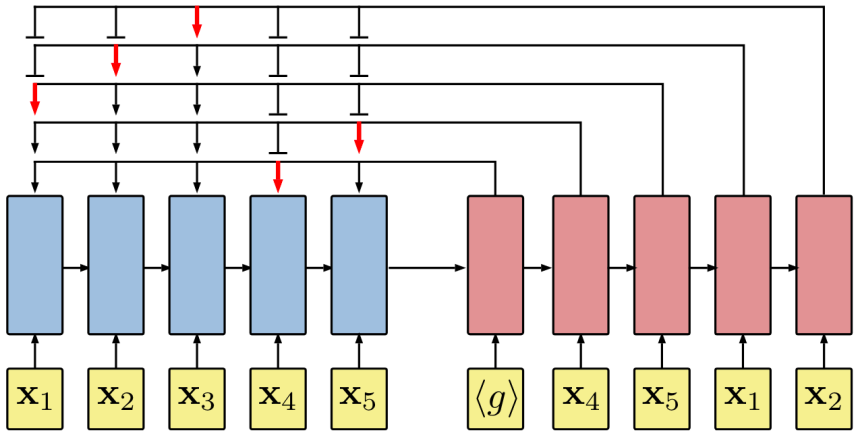
Learning to Construct

- End-to-end learning
  - **Input**: problem instance
  - **Output**: a solution
- Usually outputs the final solution, but may also be used as initial solution followed by a search process

- Key design issues:
  - Encoding of the problem instance (input)
  - Encoding of the solution (output)
  - Model architecture
  - Learning method

Example: RNN for TSP

- Problem: Find the shortest cycle to visit all the nodes in a graph
- Design:
  - Input: nodes as a sequence
  - Output: solution as a sequence
  - Model: recurrent neural network



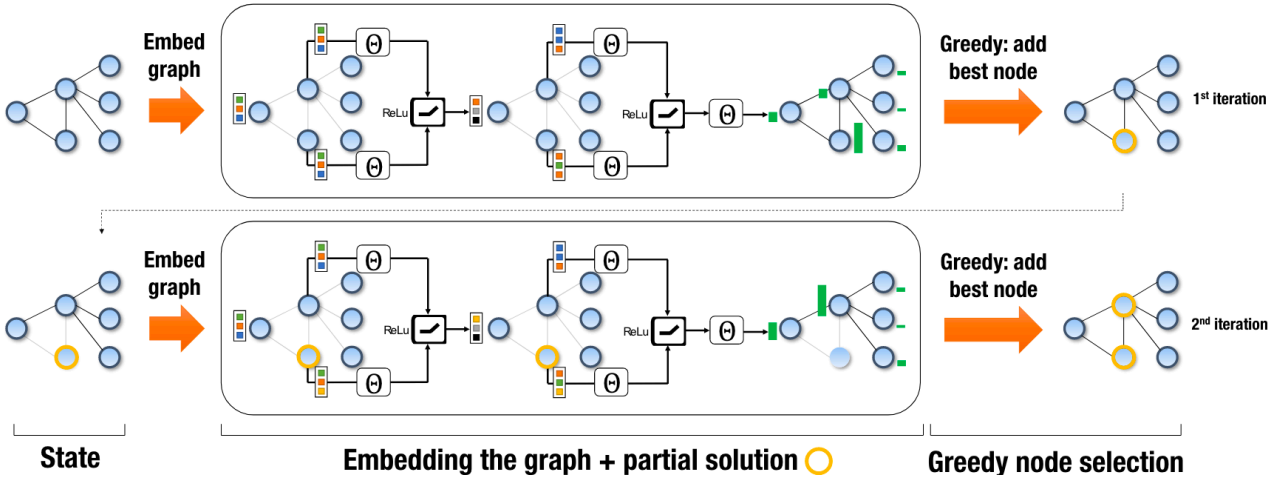
Cannot generalise to more nodes

Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*.

Example: Learn to Solve Minimum Vertex Cover

- Problem: Find the minimal subset of vertices to cover all the vertices in a graph
- Design:
  - Input: graph embedding (graph neural network)
  - Output: 0/1 for each vertex
  - Model: graph neural network
  - Reinforcement learning: learn a policy to add nodes

Can generalize to different graphs



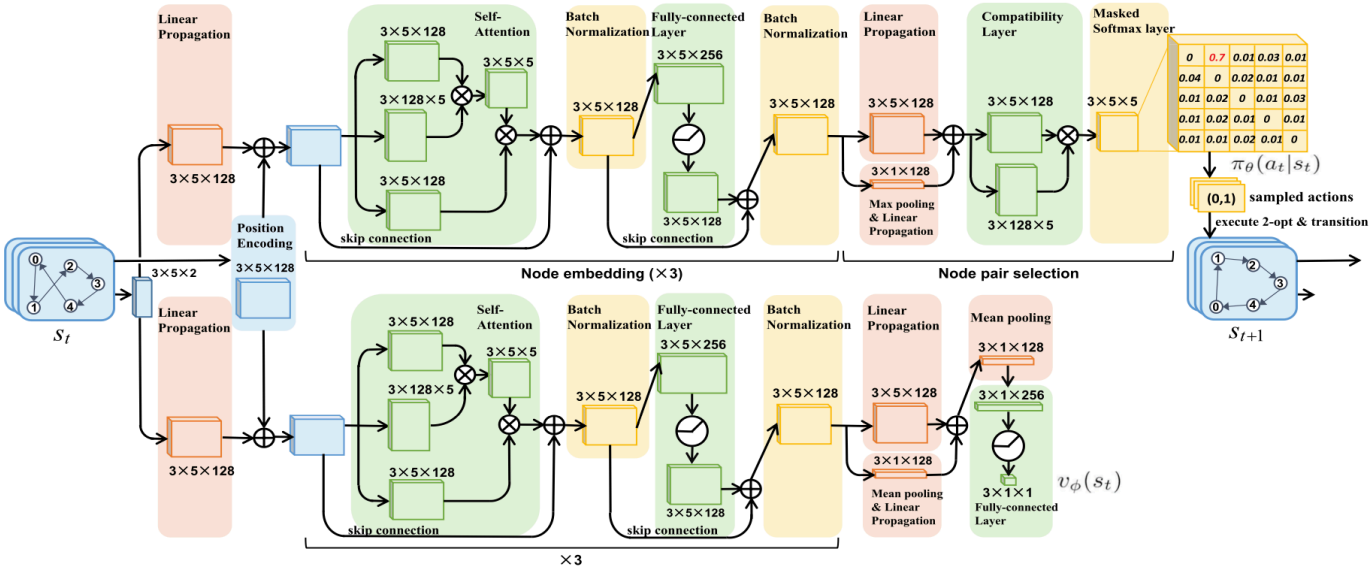
Khalil, E., Dai, H., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. Advances in Neural Information Processing Systems, 30.

Learn to Search

- Machine learning to make design decisions of search algorithms
  - Select the next branch during branch and bound
  - Select the next neighbouring solution during local search
  - Select the next crossover/mutation operator in an EC algorithm

Example: Learn 2-opt for Routing

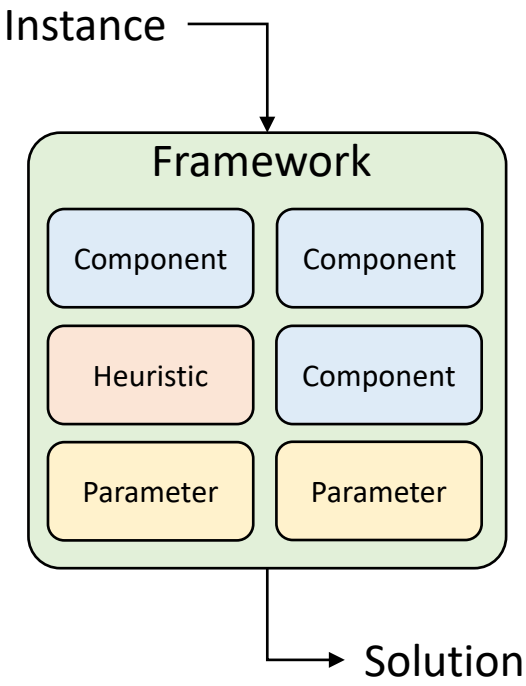
- Iterative improvement process
  - Solution -> 2-opt -> Solution -> 2-opt -> …
  - Learn a policy to select a pair of nodes to conduct 2-opt
  - Reinforcement learning process



Wu, Y., Song, W., Cao, Z., Zhang, J., & Lim, A. (2021). Learning improvement heuristics for solving routing problems. IEEE Transactions on Neural Networks and Learning Systems, 33(9), 5057-5069.

Evolutionary Computation to Learning Combinatorial Optimisation Heuristics

EC to Learn CO Heuristics



- Define a space of heuristics
  - A framework for the heuristic to work within
    - Input: a CO problem instance
    - Output: a solution
  - Representation/Encoding for heuristics
- Define the evaluation of heuristics
  - Training data
  - Performance metric (e.g., effectiveness, efficiency, complexity)
  - Black-box, non-differentiable, discrete
- Design search algorithms by EC
  - Good at black-box optimisation
  - Handle non-differential space
  - Multi-objective optimisation

EC to Learn CO Heuristics

Preprocessing

Design a **framework** Γ parameterized with the CO heuristic

Encoding

Decide the **space** $\mathcal{H}$ of all possible CO heuristics in consideration

Data:

Determine a **training set** P (problem instances) to train the heuristics

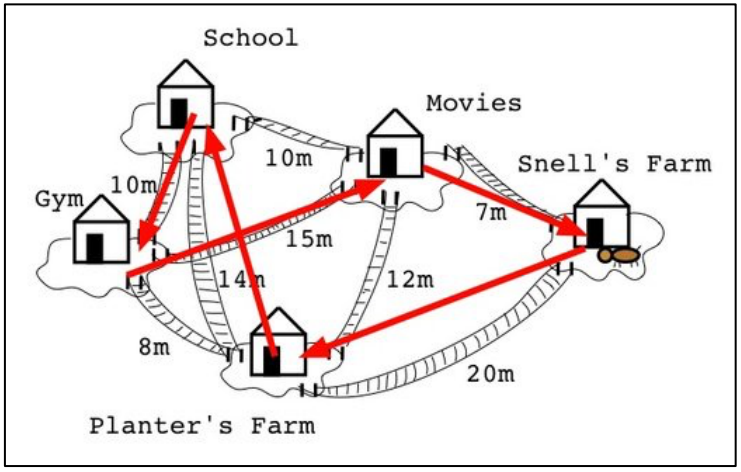
Objective:

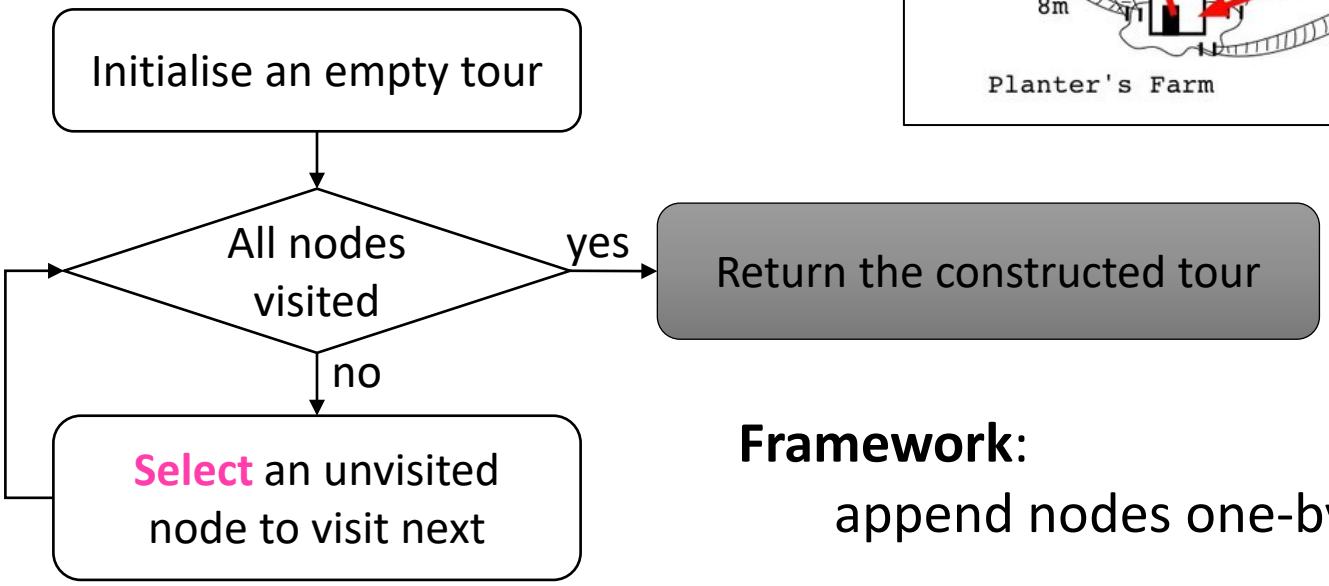
Design the training **objective function** $f(h, \mathrm{P}, \Gamma)$

Optimisation:

Find the best heuristic $h^* = \arg\min_{h \in \mathcal{H}} f(h, \mathrm{P}, \Gamma)$

Example 1: TSP





Framework:

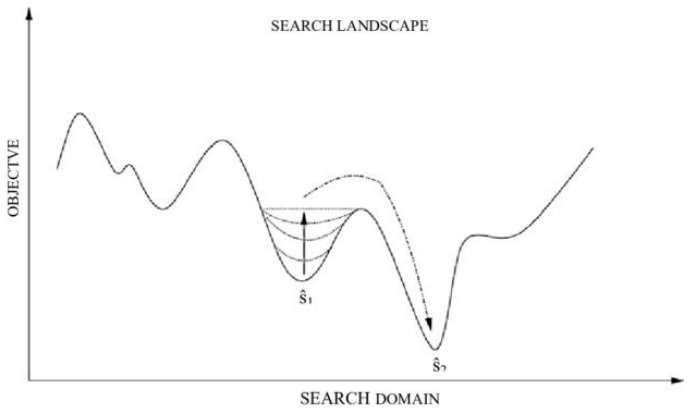
append nodes one-by-one

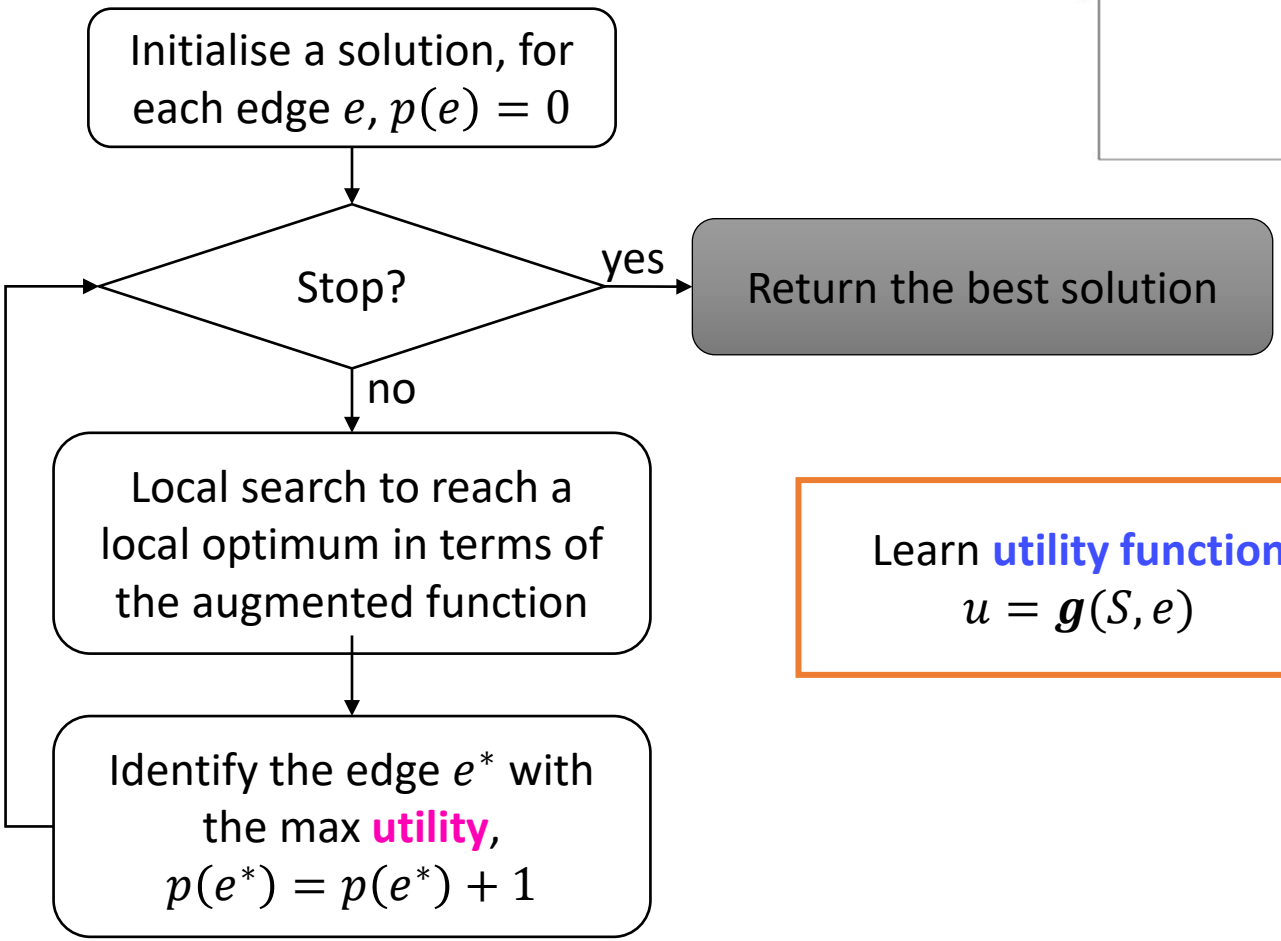
Learn heuristic for **node selection**

$$x_{next} = \boldsymbol{h}(G, X_{partial})$$

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. Advances in neural information processing systems, 30.

Example 2: GLS for VRP





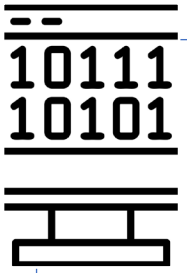
Learn **utility function**
$u = \boldsymbol{g}(S, e)$

Saining Liu, Yi Mei, Joao Costa, and Mengjie Zhang. Gpgls: Genetic programming guided local search for large-scale vehicle routing problems. In Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN). Springer, 2024

Genetic Programming for Heuristic Learning

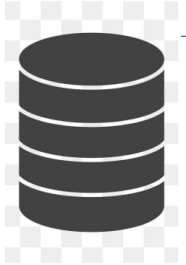
Preprocessing

- Through prior knowledge and intuition
- A **simulation** to output a solution



Encoding

- **Trees**
- **Sequence** of instructions

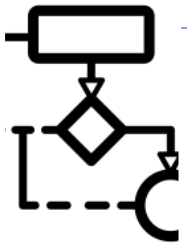


Data

- A wide range of problem instances
- Generalisation and efficiency (problem size, #instances) are important

Objective function

- Normalised objective values of the obtained solutions

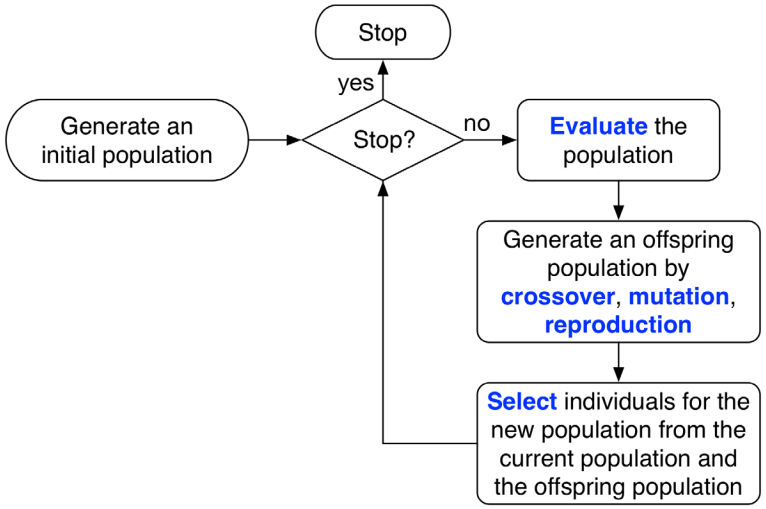


Optimisation

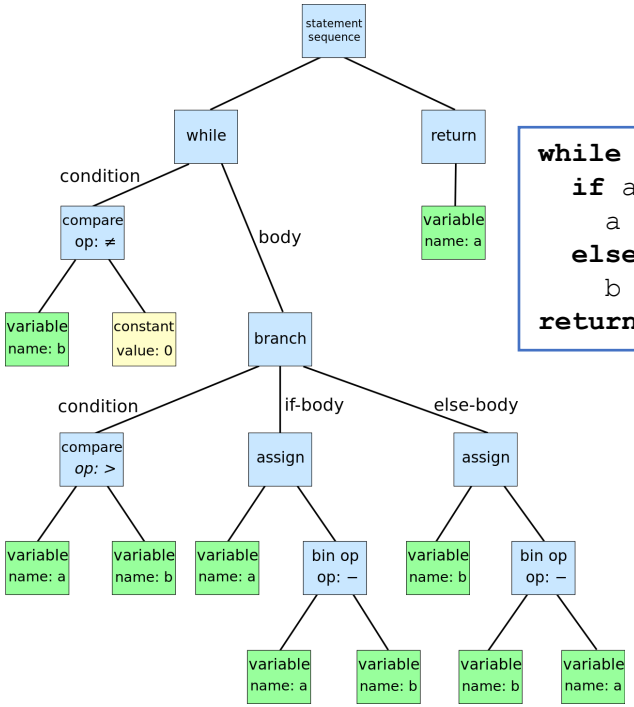
- Genetic programming

Genetic Programming (GP)

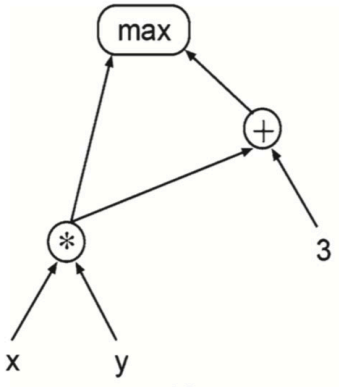
- A type of evolutionary algorithm
  - Evolve computer programs rather than solutions



- Representation of computer programs
  - Tree-like, graph-like, linear, …




```
while b ≠ 0
  if a > b
    a := a − b
  else
    b := b − a
return a
```

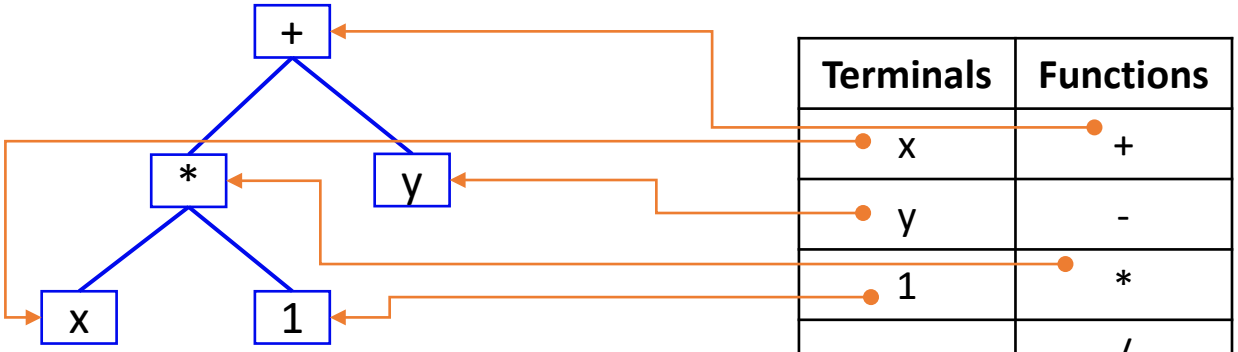


$\max(x * y, x * y + 3)$

```
r[3] = r[1] / 1.3;
r[1] = r[2] * -5.5;
r[0] = if r[1] < 0 then
          r[0] else r[3];
```

Genetic Programming (GP)

- Individual generation (Tree-based representation)
  - Terminal set: inputs of the program and constants, no argument, form the leaf nodes,
  - Function set: operators to the inputs and intermediate results of the program (e.g., +, -, max, …), form the non-leaf nodes
- Start from the **root** node
- For each node, **randomly sample from the terminal/function set**
  - If sampling from the terminal set, then stop this branch
  - If sampling from the function set, create the child nodes, and **recursively sample** the child nodes

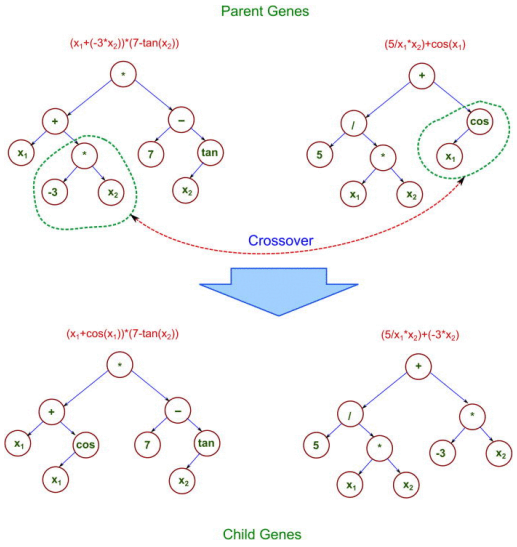


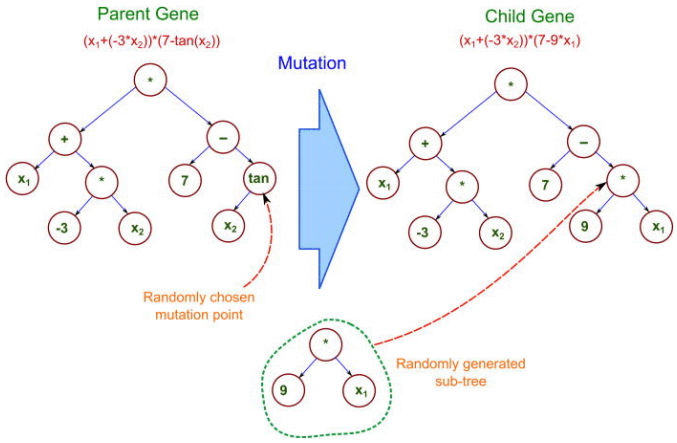
Terminals	Functions
x	+
y	-
1	*
	/

$$x * 1 + y$$

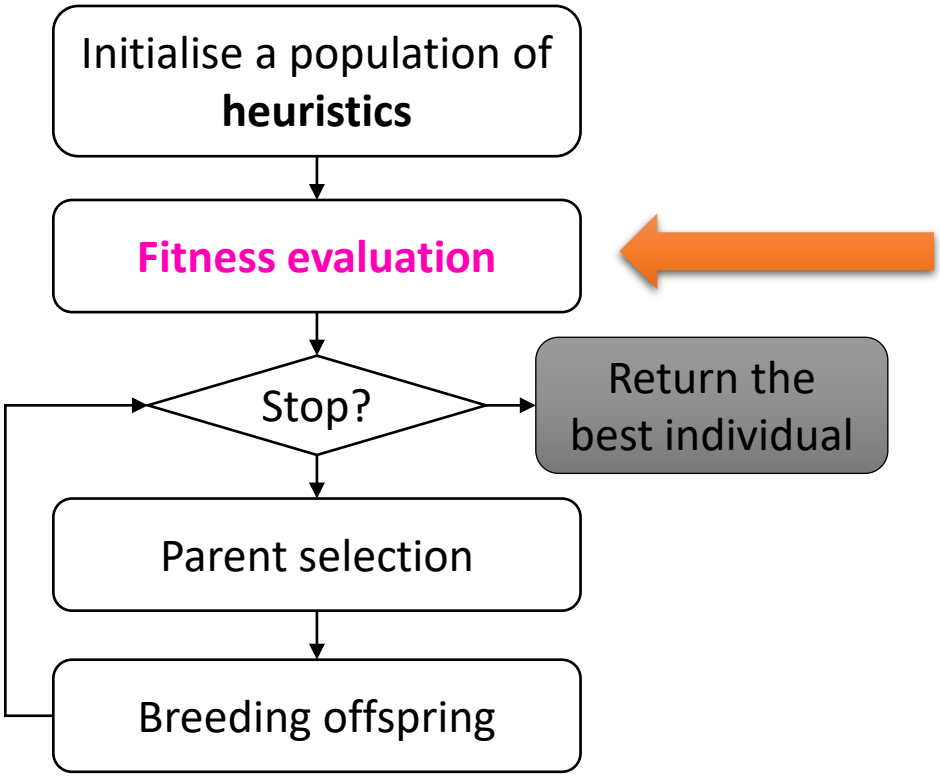
Genetic Programming (GP)

- Genetic Operators (tree-based)
  - Crossover: randomly select a sub-tree from each parent, and swap them
  - Mutation: randomly select a sub-tree from the parent, and replace the sub-tree with a randomly generated sub-tree (e.g., grow)
  - Reproduction: copy the parent directly





GP to Learn CO Heuristic



- Design a framework
- Determine a training set
- Design a fitness function

The **framework** with the **heuristic** obtains/returns a **solution** to a given **instance**

Fitness Evaluation

Heuristic + Framework

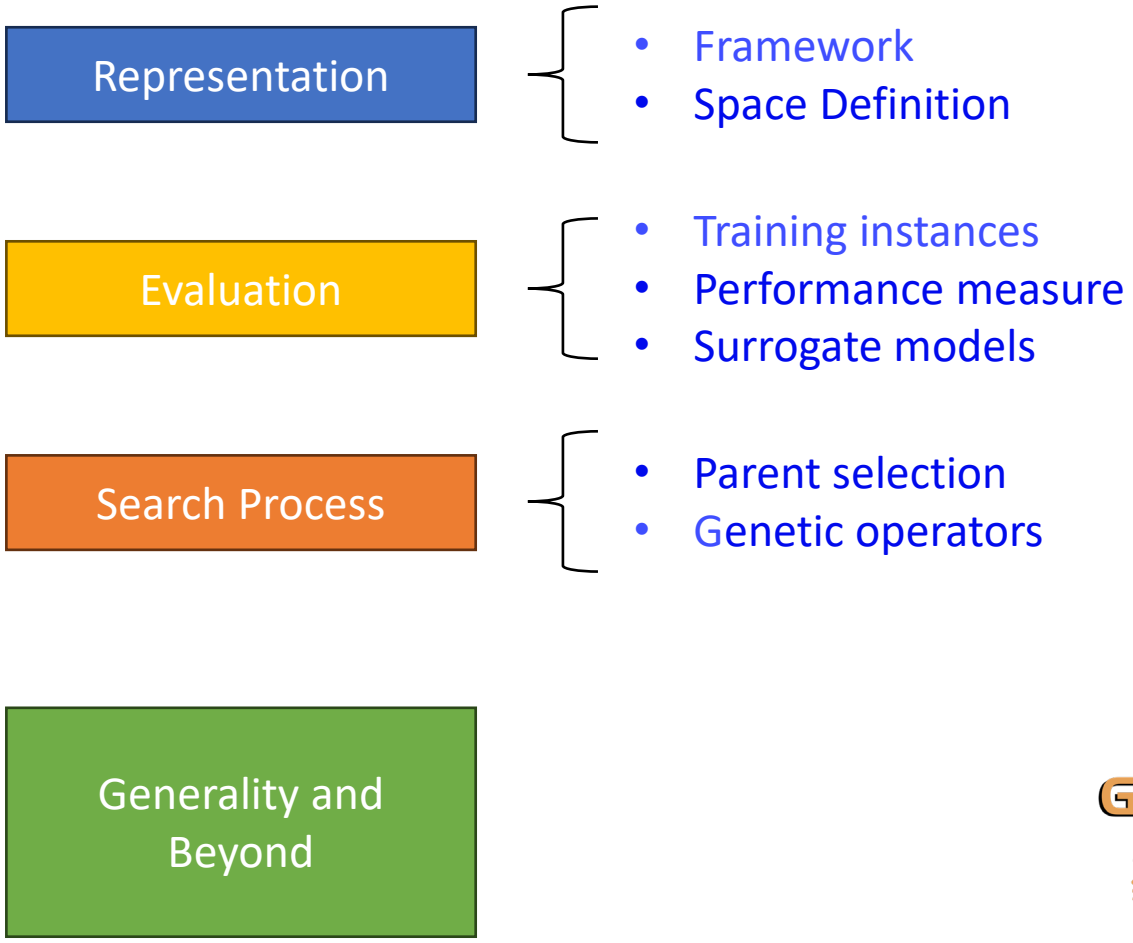
Training	Ref value		
Instance 1	120	Heuristic	Framework
Instance 2	240	Heuristic	Framework
Instance 3	180	Heuristic	Framework

→

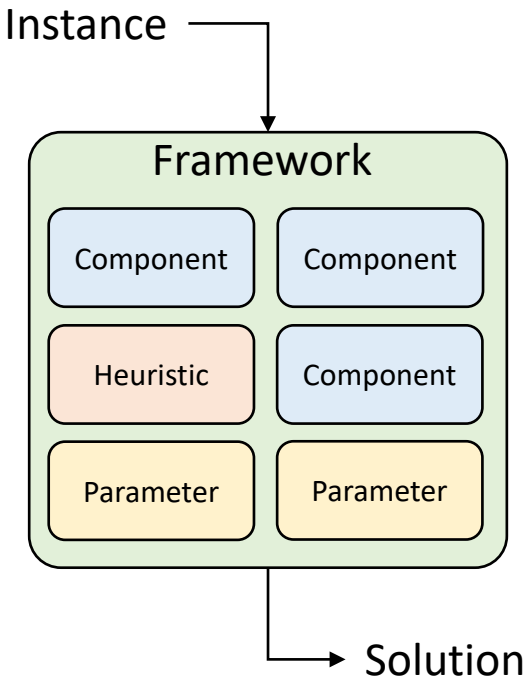
Solution	Obj
S1	140
S2	250
S3	200

$$fitness = \frac{\left(\frac{140}{120} + \frac{250}{240} + \frac{200}{180}\right)}{3} = 1.11$$

Design Questions



Representation



- Framework
  - **Input**: Instance
  - **Output**: Solution
  - **Parameters**: Heuristic to be learned
    - Constructive heuristic
    - Improvement heuristic / operator
    - Parameters
- Space Definition (Set of all possible heuristics)
  - Features / Terminals
  - Functions / Non-terminals
  - Model structure (tree-based, linear-based, …)
  - Possible combinations (Grammar)

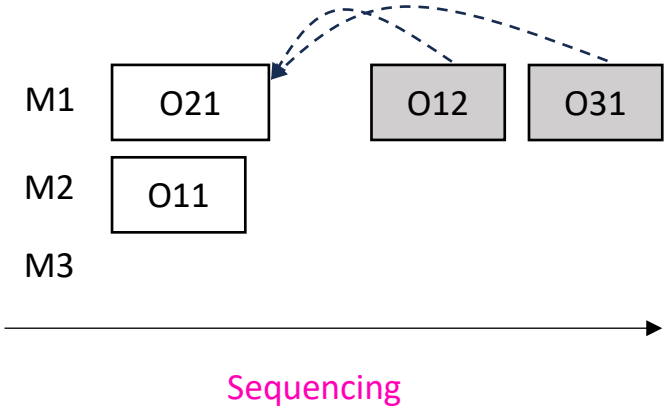
Framework

- The framework is usually designed based on intuition and domain knowledge
    1. Take an existing algorithm: everything is manually designed and hard coded
    2. Identify the component(s) of the existing algorithm to be learned
    3. Parametrise the identified component(s)

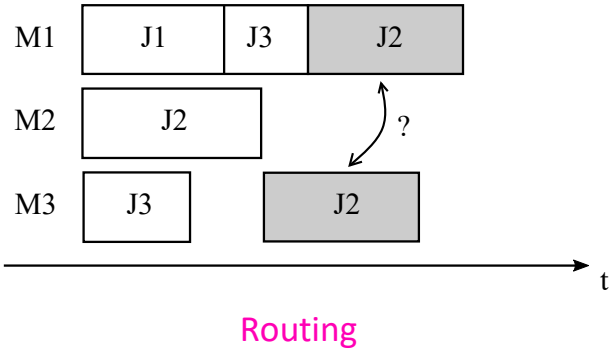
- Example
    - Learn flexible job shop scheduling heuristics

Flexible Job Shop Scheduling Heuristics

- Process set of jobs by a set of machines
- Each job has a sequence of operations
- Each operation can be processed by multiple candidate machines
- Routing **decision**: select a machine for an operation
- Sequencing **decision**: select the next operation from a machine's queue



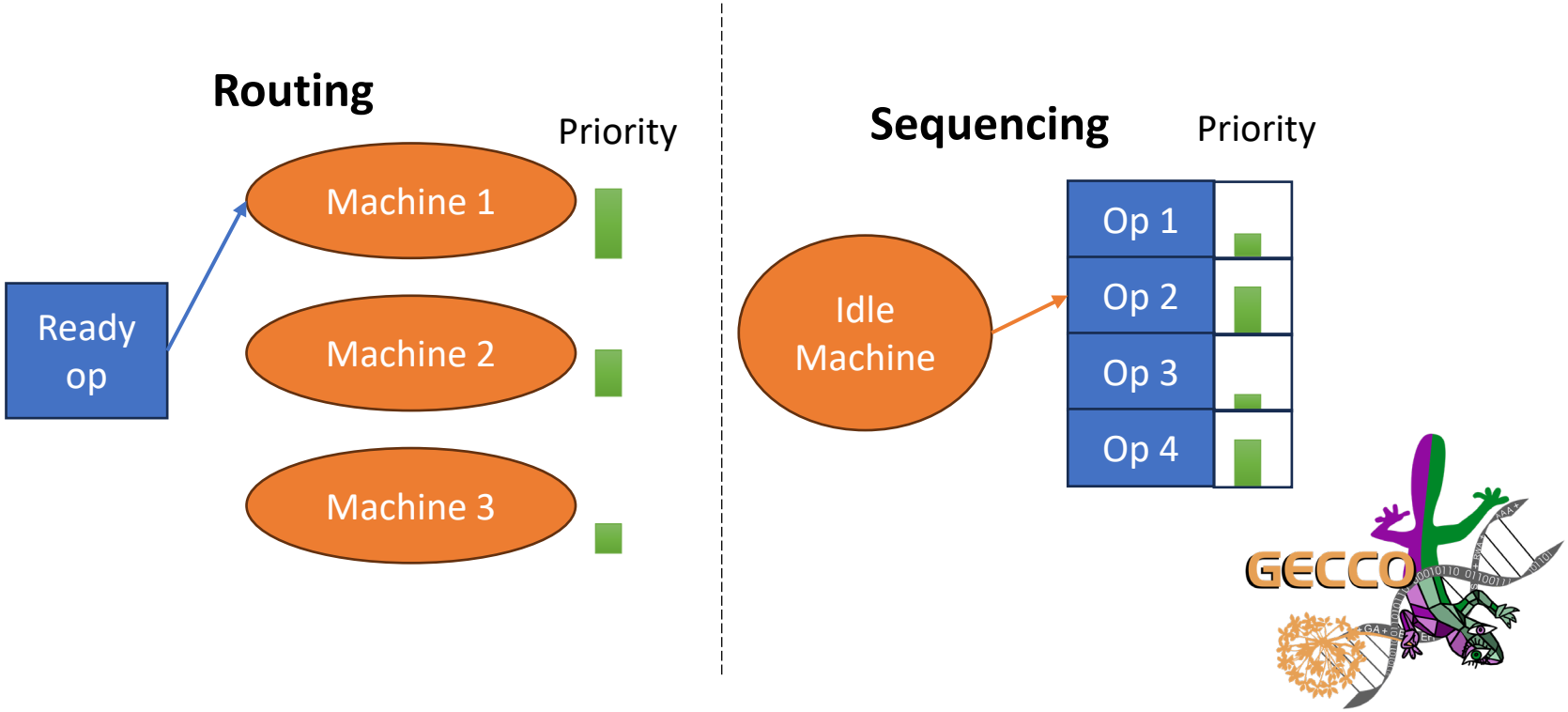
Sequencing



Routing

Framework 1: non-delay

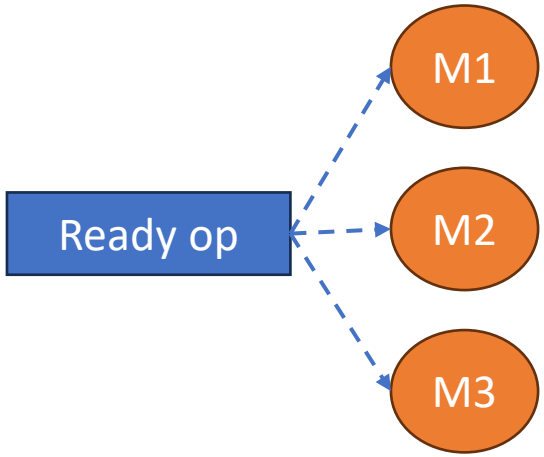
- Select machine as soon as an operation is ready
- Select operation as soon as a machine becomes idle



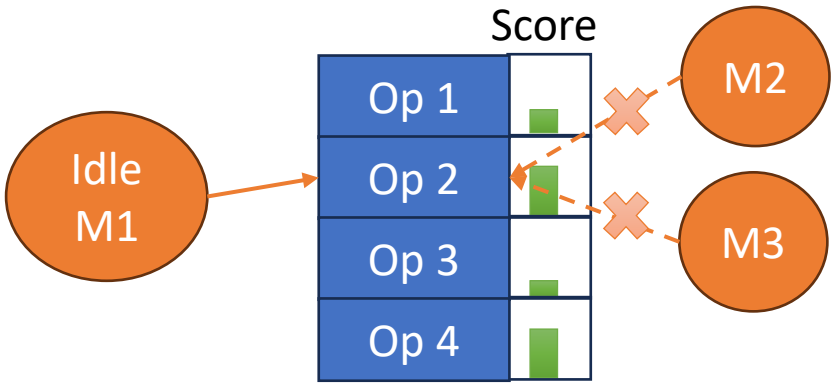
Daniel Yska, Yi Mei, and Mengjie Zhang. Genetic programming hyper-heuristic with cooperative coevolution for dynamic flexible job shop scheduling. In Proceedings of the European Conference on Genetic Programming (EuroGP), pages 306--321. 2018

Framework 2: delayed-routing

- Routing (machine selection) is delayed until at least one candidate machine becomes idle



No Routing:
no machine is idle



Sequencing
(+ delayed routing)

Binzi Xu, Yi Mei, Yan Wang, Zhicheng Ji, and Mengjie Zhang. Genetic programming with delayed routing for multi-objective dynamic flexible job shop scheduling. Evolutionary Computation, 29(1):75--105, apr 2021

Space Definition: Terminals

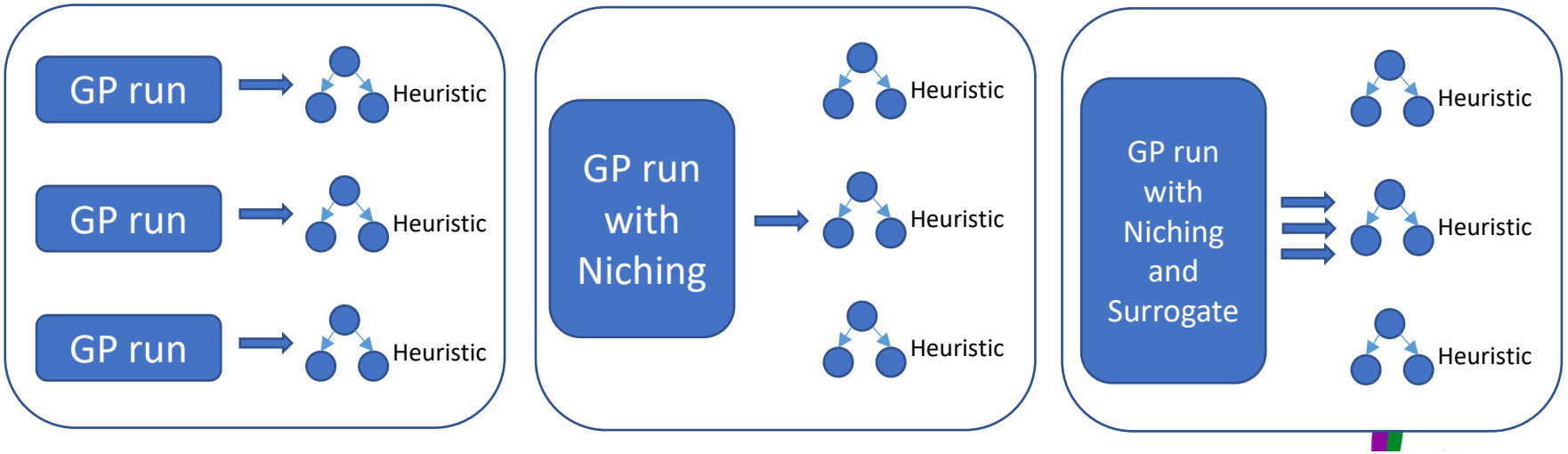
- Consider all the features that might affect the decisions
- But this will lead to a huge GP search space
- Can alleviate by feature/terminal selection

- Q: How do we know which features are important?
- A: If a feature is used by many different good heuristics

- But we don't have many different good heuristics

Space Definition: Terminals

1. Collect a diverse set of good heuristics
2. Calculate the importance of each terminal to each heuristic
3. Select terminals by aggregated importance



[Intuitive]: Several independent GP runs, gather the best-of-run heuristics

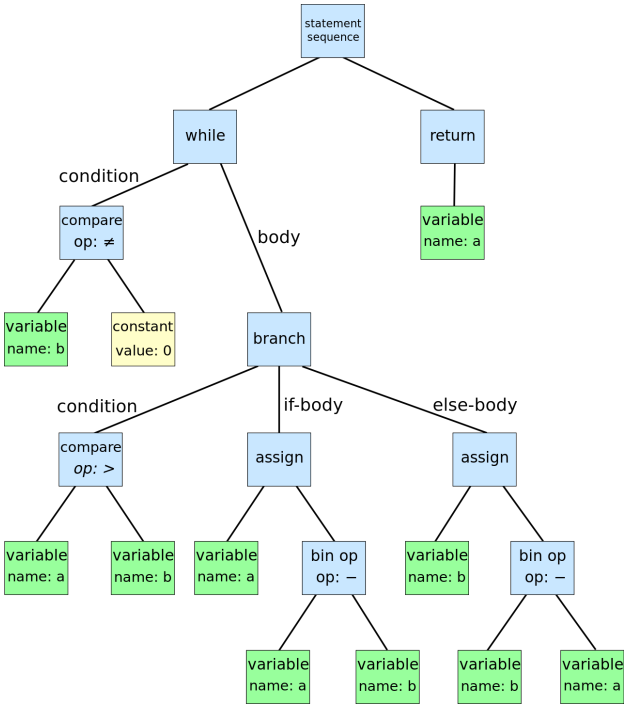
[Better]: A single GP run with niching, gather the diverse local optimal heuristics

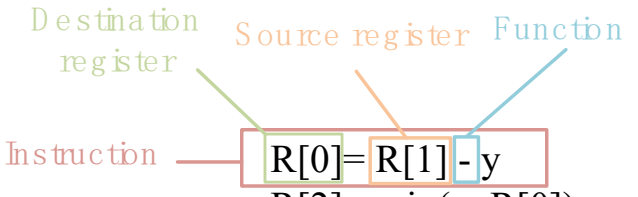
[Best]: A single GP run with niching, and surrogate evaluation

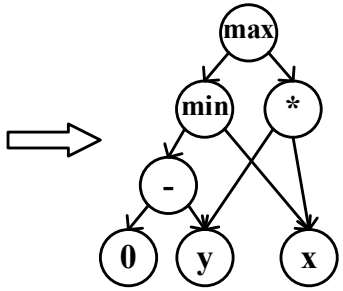
- Yi Mei, Mengjie Zhang, and Su Nyugen. Feature selection in evolving job shop dispatching rules with genetic programming. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pages 365--372. ACM, 2016.
- Yi Mei, Su Nguyen, Bing Xue, and Mengjie Zhang (2017). An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming. IEEE Transactions on Emerging Topics in Computational Intelligence, 1(5), 339-353.

Space Definition: Model Structure

- Tree-based structure
  - Abstract syntax tree of program
- Linear-based structure
  - A sequence of register-based instructions
  - Can represent a Directed Acyclic Graph (DAG)
  - Easy reuse of building blocks

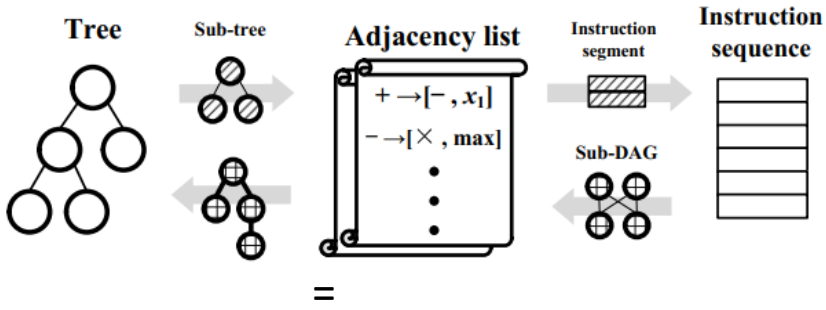




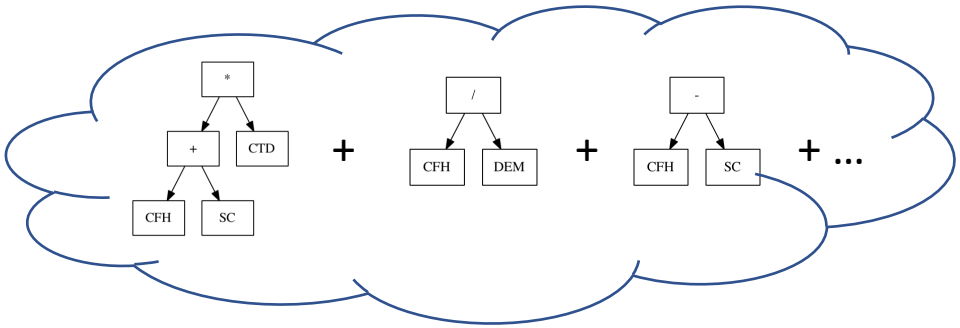


Space Definition: Model Structure

- Using tree-based and linear-based representation simultaneously
  - Multiple sub-populations, each with a different representations
  - Interact with crossover
  - Adjacency list as intermediate

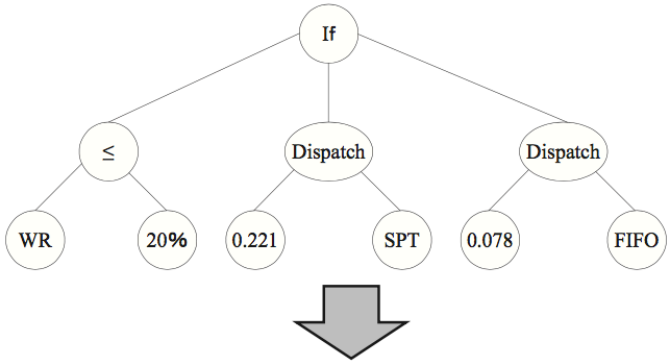


- Multi-tree / Ensemble representation
  - Each tree stands for a different heuristic (e.g., routing and sequencing rules)
  - Trees collaborate to make decision (e.g., voting)



Space Definition: Grammar

- Restrict how features/terminals can be combined
  - Smaller search space, better interpretability
  - Disconnected space, may lose effectiveness
- **Example**: grammar for meaningful dispatching rules
  - Recursive depth-first sampling, generate a tree




```
If (workload ratio is less than or equal to 20%)
      Use the SPT rule with non-delay factor of 0.221
Else
      Use the FIFO rule with non-delay factor of 0.078
```

```
Start ::= <action>
<action> ::= <if> | <dispatch>
<if> ::= if <attributetype><op>  <threshold>
then <action> else <action>
<op> ::= ≤ | >
<attributetype>::=  WR | MP | DJ | CMI | CWR | BWR
<threshold> ::=10%|20%|30%|40%|50%|60%|70%|80%|90%|100%
<dispatch> ::= assign <nondelayfactor>assign  <rule>
<nondelayfactor>::=  uniform[0,1]
<rule> ::=FIFO | SPT | LPT | LSO | LRM | MWKR | SWKR |
MOPR | EDD | MS | WSPT
```

Nguyen, S., Zhang, M., Johnston, M. and Tan, K.C., 2013. A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. *IEEE Transactions on Evolutionary Computation*, *17*(5), pp.621-639.

Space Definition: Grammar

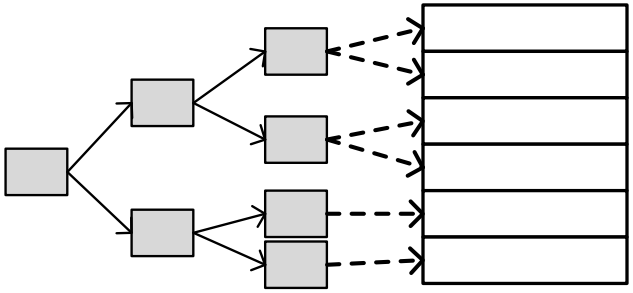
- Linear GP can also have grammar!
  - Module Context-Free Grammar for sequence of instructions

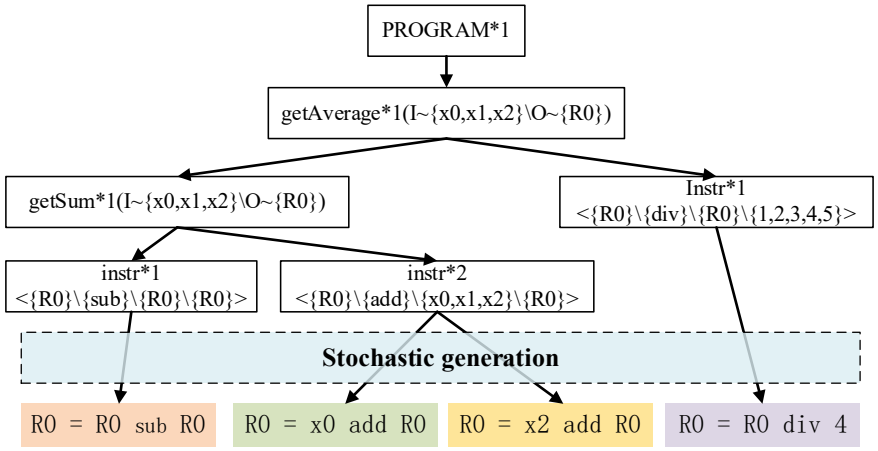
Grammar rules based on domain knowledge

```
defset FUNS {add,sub,mul,div};
defset REG {R0,R1,R2,R3,R4};
defset INPUT {x0, x1, x2};
defset CONSTANTS {1,2,3,4,5};

getSum(I\O) ::= <O\{sub}\O\O>::<O\{add}\I\O>*5;

getAverage(I\O) ::= getSum(I~I\O~O)::<O\{div}\O\{CONSTANTS}>;

getVariance(I\O) ::=getAverage(I~I\O~{R4})::(<{R1,R2,R3}\{sub}\{INPUT}\{R4}>
::<{R1,R2,R3}\{mul}\{R1,R2,R3}\{R1,R2,R3}>)*3::getAverage(I~{R1,R2,R3}\O~O);

PROGRAM ::= getSum(I~{INPUT}\O~{R0}) | getAverage(I~{INPUT}\O~{R0})
| getVariance(I~{INPUT}\O~{R0});
```

Derivation tree based on grammar

LGP program based on derivation tree





Z. Huang, Y. Mei, F. Zhang, and M. Zhang, "Grammar-guided Linear Genetic Programming for Dynamic Job Shop Scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Jul. 2023, pp. 1137–1145. **[GP Track Best Paper Award]**

Evaluation

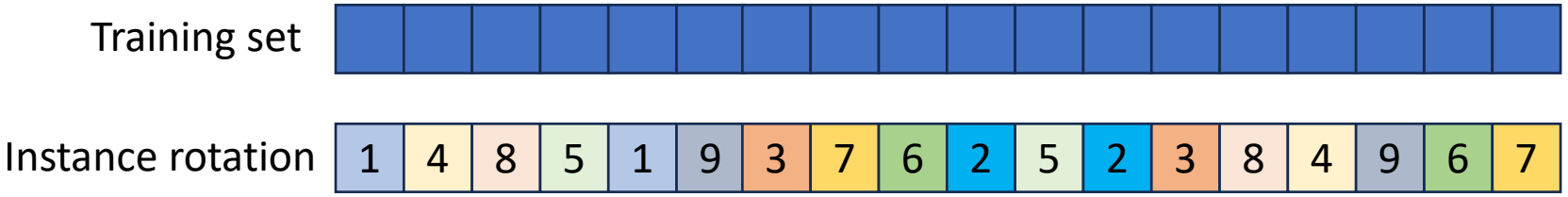
- Training instances
  - A wide range of training instances is needed for generalization
  - However, too many training instances will slow down the evaluation and waste resources
  - Generalisation vs Efficiency

- Performance metrics
  - Effectiveness (e.g., quality of obtained solutions)
  - Efficiency (computational complexity)
  - Interpretability
  - …

- Surrogate models

Training Instances

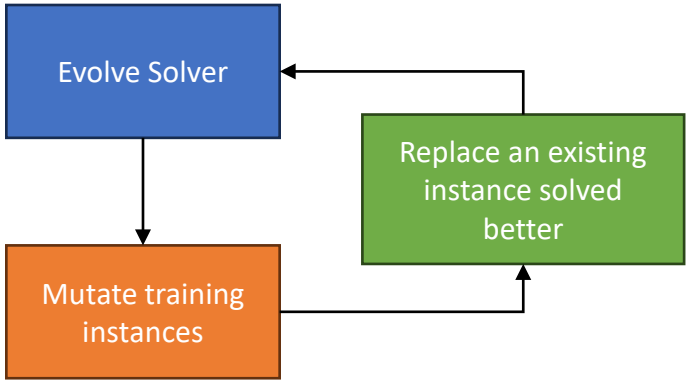
- A wide range of training instances + instance rotation
  - Similar to batch learning
  - Split the training instances into small subsets (batches)
  - At each generation, use a different subset (same for all individuals in the population)

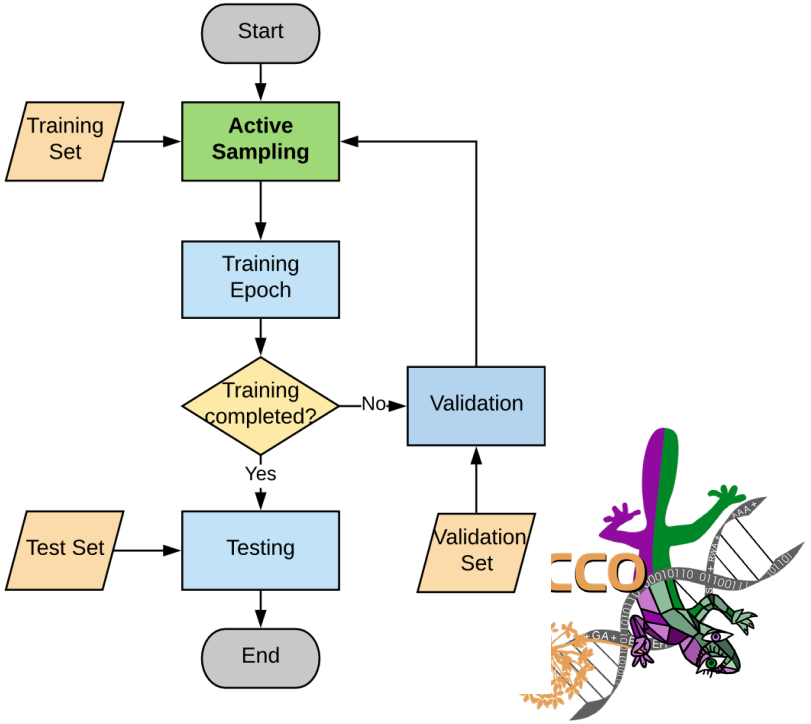


Hildebrandt, T., Heger, J., & Scholz-Reiter, B. (2010, July). Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation* (pp. 257-264).

Training Instances

- Intelligently select/sample the best training instances
  - Remove the instances that have been solved well
  - Add/Retain the instances that are not solved
  - Focus on the instances that best distinguish good/bad heuristics





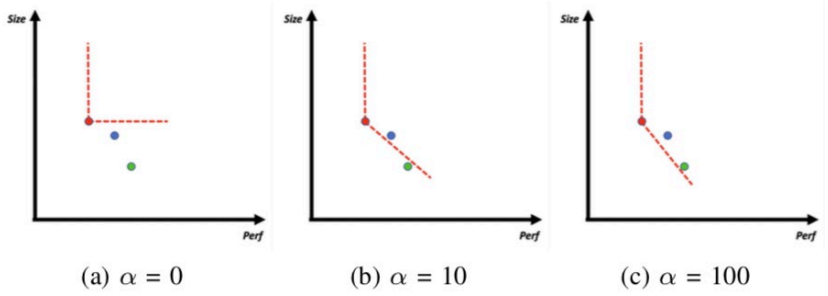
- Tang, K., Liu, S., Yang, P., & Yao, X. (2021). Few-shots parallel algorithm portfolio construction via co-evolution. IEEE Transactions on Evolutionary Computation, 25(3), 595-607.
- Karunakaran, D., Mei, Y., Chen, G., & Zhang, M. (2019, June). Active sampling for dynamic job shop scheduling using genetic programming. In *2019 IEEE Congress on Evolutionary Computation (CEC)* (pp. 434-441). IEEE.

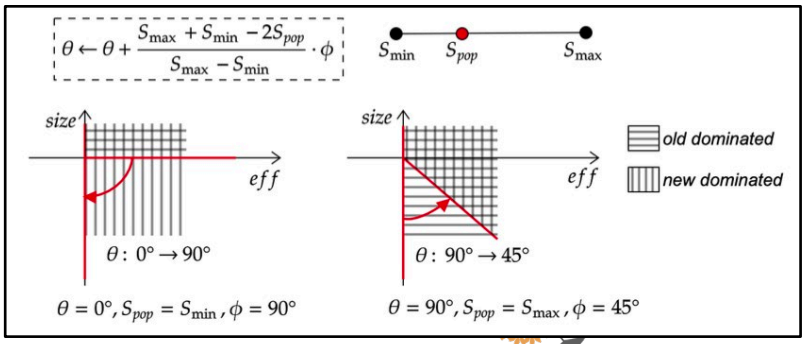
Performance Metrics

- Average/Normalised solution quality obtained over the training/validation set (Effectiveness)
- For iterative improvement frameworks, when to stop?
  - After converge, balance between effectiveness and efficiency
  - During training, as early as possible if can distinguish good and bad heuristics
- For stochastic framework/heuristic, need multiple runs for statistical significance (efficiency issue)
  - Learn deterministic algorithms (e.g., ϵ greedy)
- Model size/complexity, interpretability
- Multi-objective optimisation

Effectiveness vs Interpretability

- MOGP: f1 = effectiveness; f2 = model size (interpretability proxy)
- It is hard to balance effectiveness (e.g., accuracy) and size during the MOGP search
- If not evolve properly, the population can be easily biased to small but bad individuals, and lose exploration ability
- Use α-dominance to adjust the balance between effectiveness and size
  - $\alpha = 0$: normal dominance relationship
  - $\alpha = \infty$: single objective with only effectiveness
- Adapt α during the search

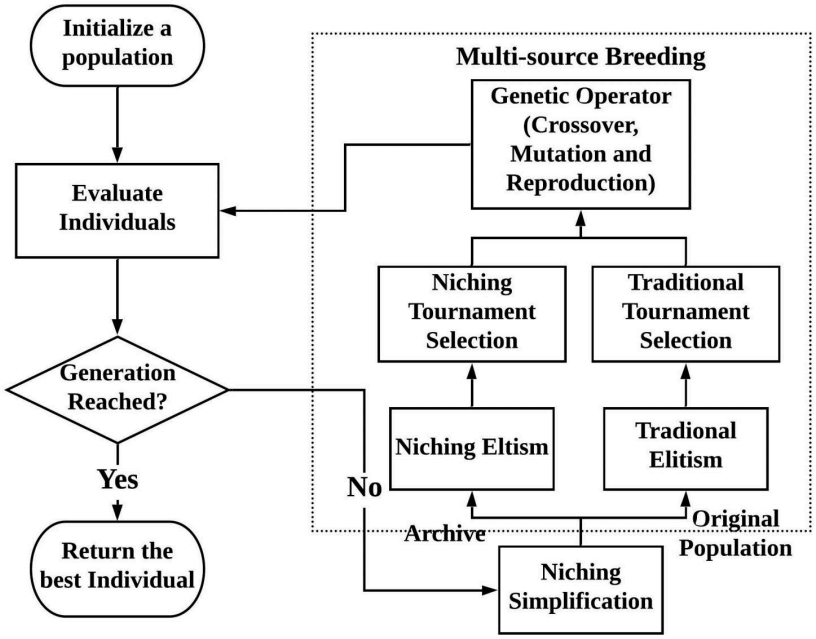


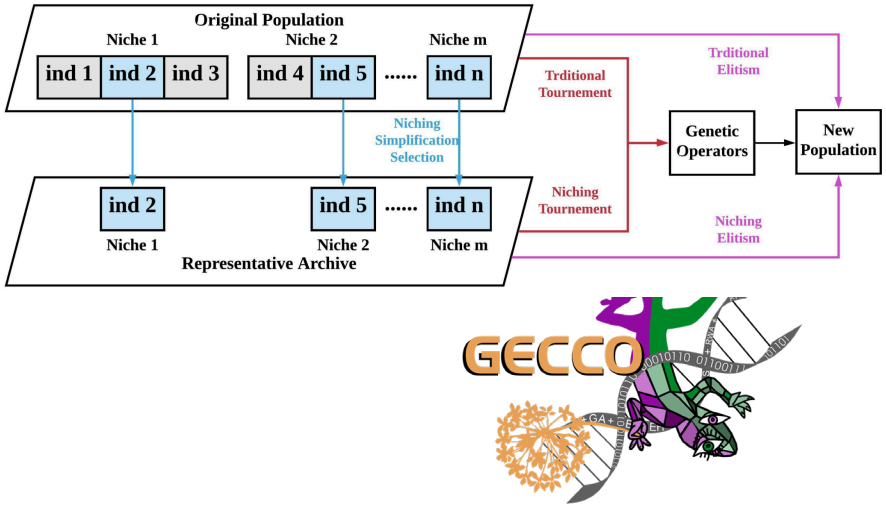


Shaolin Wang, Yi Mei, and Mengjie Zhang. A multi-objective genetic programming algorithm with αdominance and archive for uncertain capacitated arc routing problem. IEEE Transactions on Evolutionary Computation, jul 2022

Effectiveness vs Interpretability

- Niching GP
- Group individuals into niches, each niche contains individuals with the same fitness value
- Niching selection: select the smallest individual from each niche
- Multi-source breeding

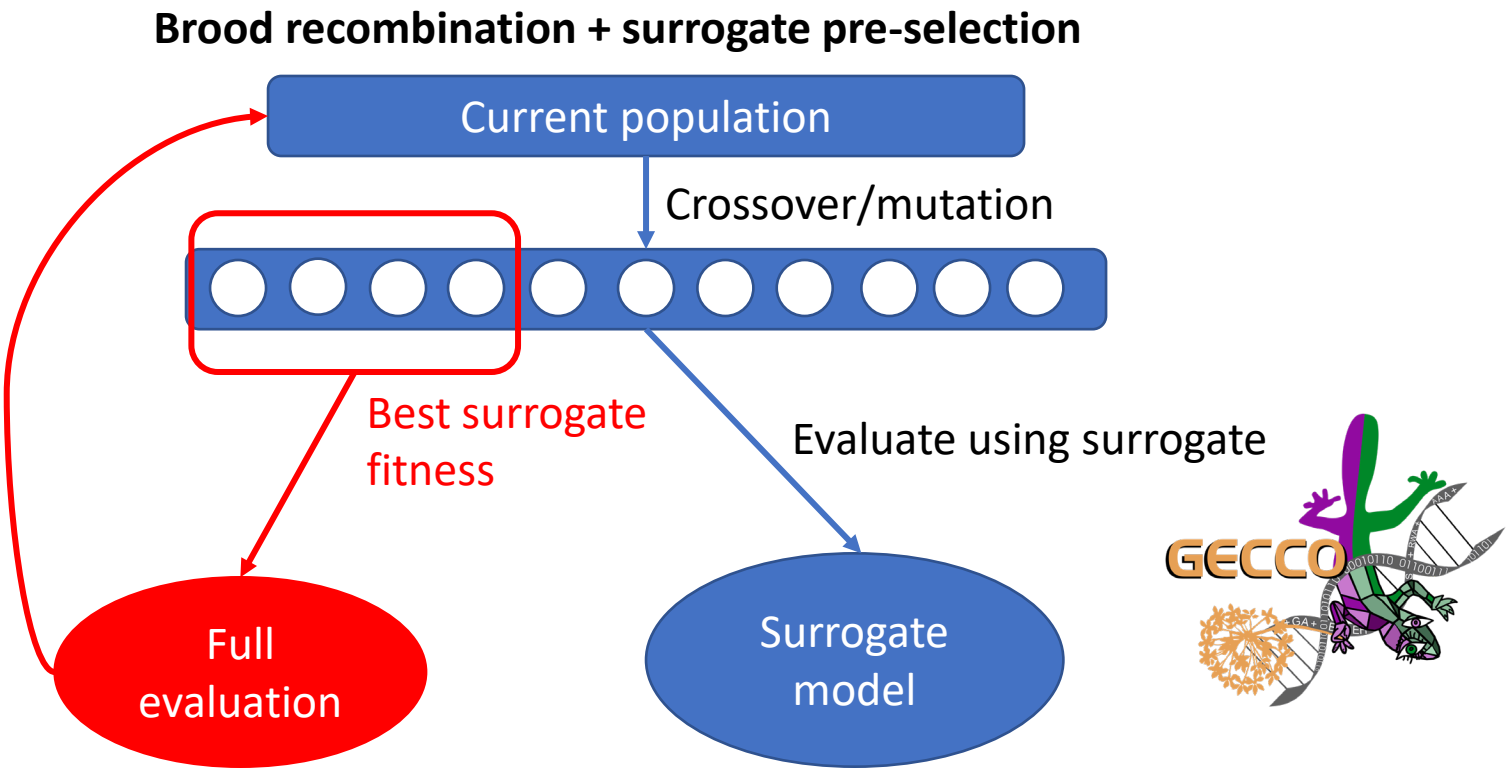




Shaolin Wang, Yi Mei, Mengjie Zhang, and Xin Yao. Genetic programming with niching for uncertain capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation*, 26(1):73--87, feb 2022

Surrogate Models

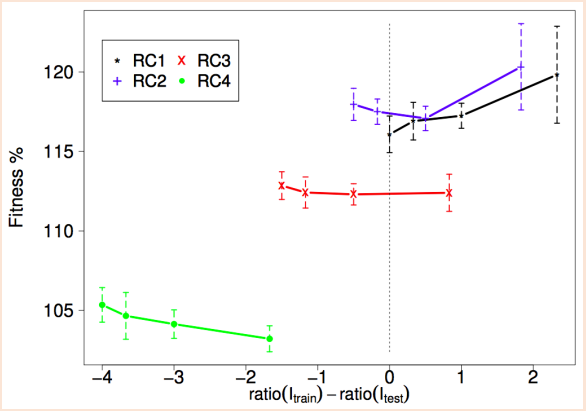
- Brood recombination: generate a large number of offspring
- Surrogate pre-selection: use surrogate to predict fitness and select the best parts

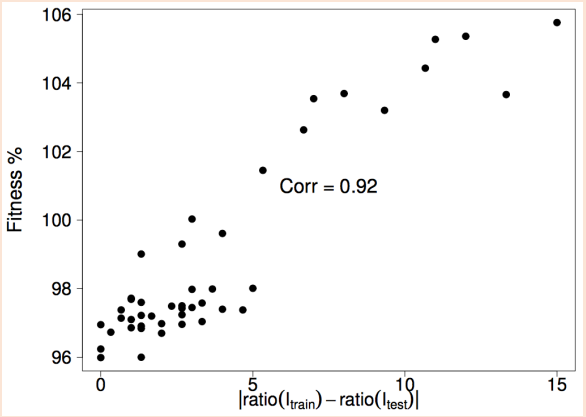


Surrogate Models

- Small instances with careful configurations, faster and correlated

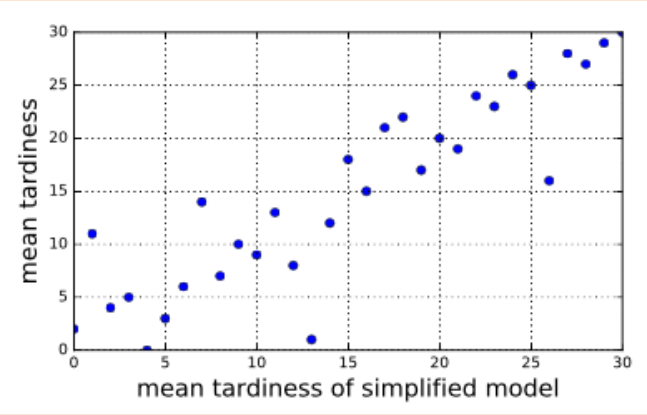
Static JSS





Dynamic JSS

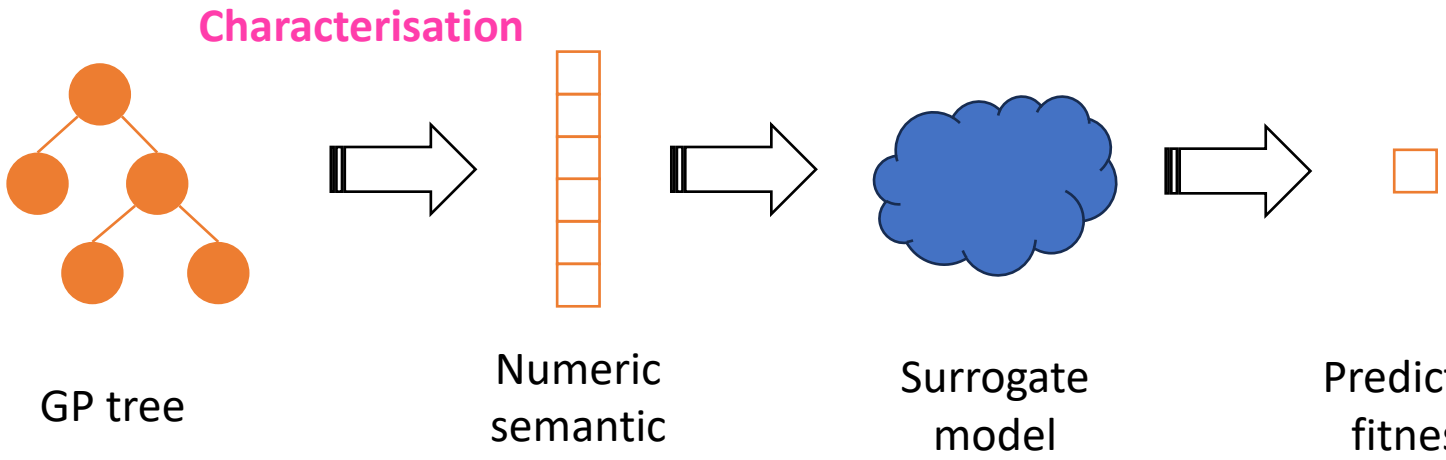
Configuration	Original	Surrogate
No. Machines	10	5
No. Jobs	5000	500
No. Warmup Jobs	500	100
Min Ops	2	2
Max Ops	14	7



- Yi Mei, Mengjie Zhang, "A Comprehensive Analysis on Reusability of GP-Evolved Job Shop Dispatching Rules," *IEEE World Congress in Computational Intelligence (WCCI)*, Vancouver, Canada, 2016.
- Nguyen, S., Zhang, M. and Tan, K.C., Surrogate-Assisted Genetic Programming With Simplified Models for Automated Design of Dispatching Rules. IEEE Transactions on Cybernetics, DOI 10.1109/TCYB.2016.2562674.

Surrogate Models

- Semantic/Phenotypic Characterisation
  - Convert a tree structure to a numeric vector



Hildebrandt, T. and Branke, J., 2015. On using surrogates with genetic programming. *Evolutionary computation*, *23*(3), pp.343-367.

Surrogate Models

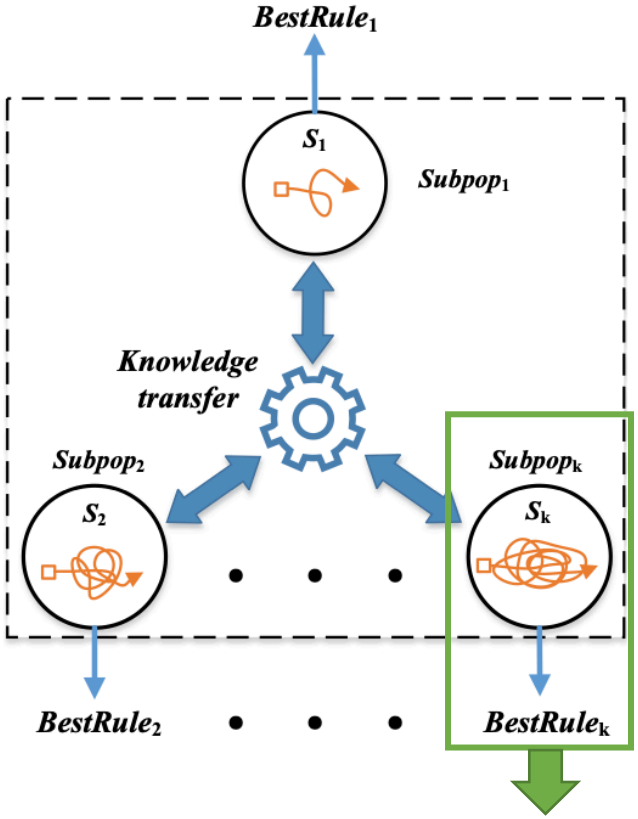
- An example in learning dispatching rules for dynamic job shop scheduling
  - A set of decision situations and a reference rule
  - For each decision situation, measure the difference between the reference rule and the characterised rule
  - Characterised by a decision vector

decision situation	attribute set **s** s_1	s_2	s_3	ranking by reference rule	ranking by other rule	decision vector **d**
1	3	4	8	1	2	
1	7	6	15	**2**	1	**2**
2	23	17	1	2	2	
2	8	9	3	**3**	1	**3**
2	6	4	6	1	3	
⋮		⋮		⋮	⋮	⋮
k	7	3	9	2	2	
k	4	8	6	**1**	1	**1**

Hildebrandt, T. and Branke, J., 2015. On using surrogates with genetic programming. *Evolutionary computation*, *23*(3), pp.343-367.

Surrogate Models

- One can have multi-fidelity surrogate models
  - Different problem size, model complexity, …
  - Accuracy vs Speed
- Use multiple sub-populations
  - Each uses a different surrogate (different fidelities)
  - Sub-populations transfer knowledge by sharing parents to generate offspring
- One sub-population uses the original model (highest fidelity) and returns the final individual



Fangfang Zhang, Yi Mei, Su Nguyen, and Mengjie Zhang. "Collaborative Multi-fidelity Based Surrogate Models for Genetic Programming in Dynamic Flexible Job Shop Scheduling". *IEEE Transactions on Cybernetics*, Doi: 10.1109/TCYB.2021.3050141.

Search Process

- Parent Selection
  - Commonly used: tournament selection
  - Lexicase selection for better diversity
- Genetic Operators

Lexicase Selection

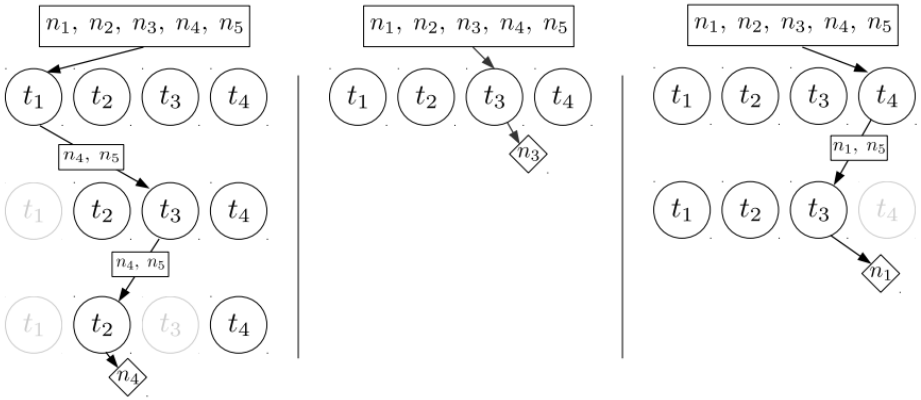
- Find specialists instead of generalists
  - Multiple cases that compose the fitness function
  - Randomly iterate through cases
  - For each case, only keep the individuals with the best case-fitness
  - Stop if only one individual retains
- Case definition
  - Instances:

$$fitness = \sum_{i=1}^{N} \left(y_i - g(\boldsymbol{x}_i)\right)^2$$

Case-fitness

$$fitness = \sum_{i=1}^{N} obj(\Gamma(h, i))$$

Case-fitness



- Helmuth, T., Spector, L., & Matheson, J. (2014). Solving uncompromising problems with lexicase selection. *IEEE Transactions on Evolutionary Computation*, *19*(5), 630-643.
- Xu, M., Mei, Y., Zhang, F., & Zhang, M. (2023). Genetic programming with lexicase selection for large-scale dynamic flexible job shop scheduling. IEEE Transactions on Evolutionary Computation.

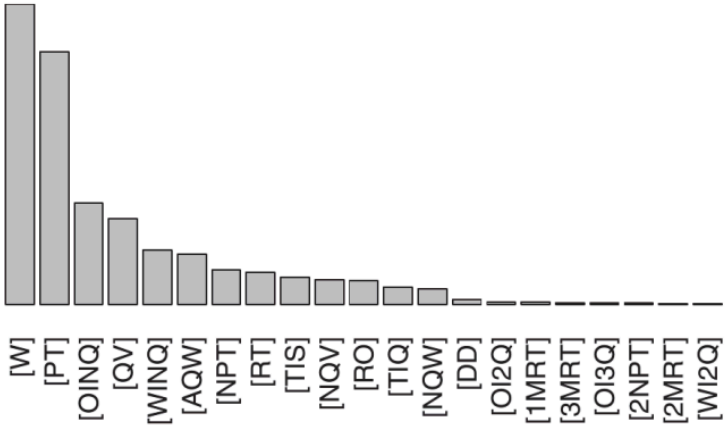
Genetic Operators

- Completely random
- Adaptive during the search process
  - Adjust the probability distribution of sampling nodes in mutation
  - In crossover, replace less important sub-trees with more-important sub-trees

Adaptive Mutation

- Adjust probability of terminals based on historical occurrences
  - A terminal is sampled with a higher probability if it occurs more often in individuals with better fitness
- After each generation,

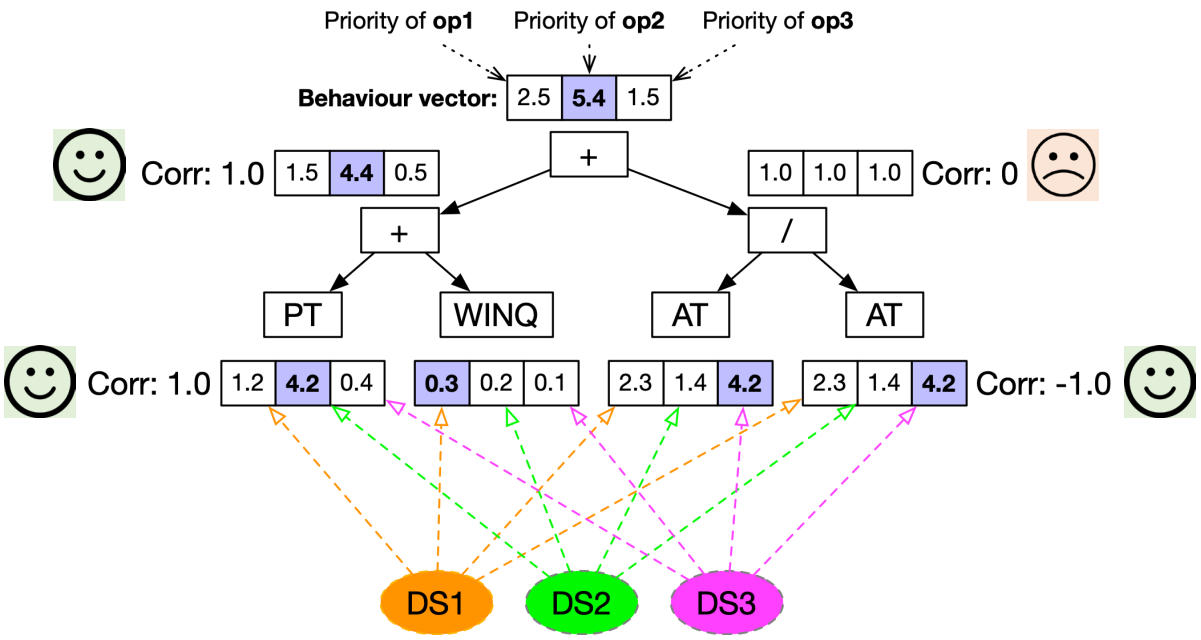
$$w_i(t) = w_i(t-1) + \sum_{j=1}^{P} \frac{count(i,j) \times fitness_j}{\sum_{k=1}^{n} count(i,j)}$$

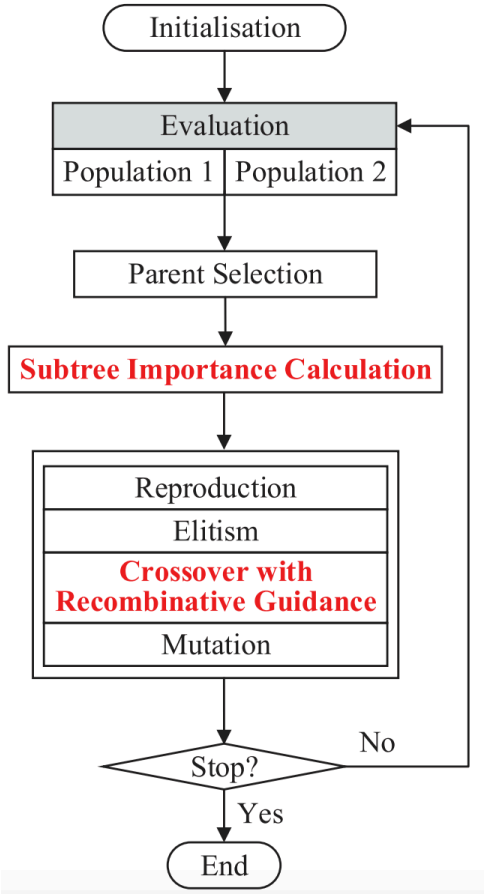


Riley, M., Mei, Y., & Zhang, M. (2016, July). Improving job shop dispatching rules via terminal weighting and adaptive mutation in genetic programming. In *2016 IEEE Congress on Evolutionary Computation (CEC)* (pp. 3362-3369). IEEE.

Adaptive Crossover

- Contribution of a sub-tree to the individual based on behaviour consistency (correlation)
- Less contribution will be more likely to be replaced

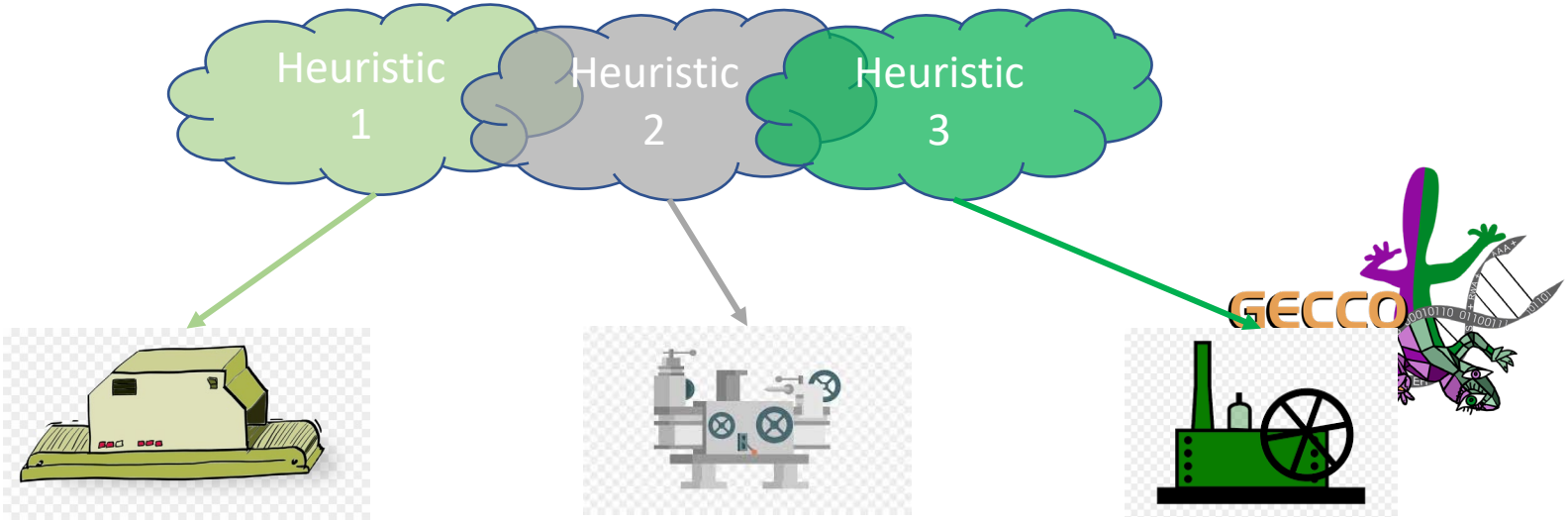




Zhang, F., Mei, Y., Nguyen, S., & Zhang, M. (2021). Correlation coefficient-based recombinative guidance for genetic programming hyperheuristics in dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation*, *25*(3), 552-566.

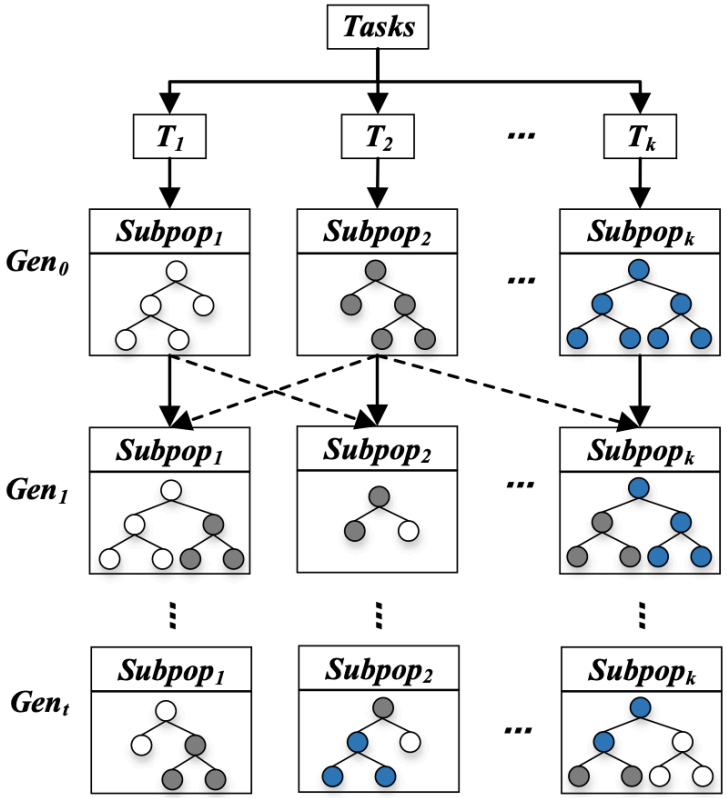
Generality and Beyond

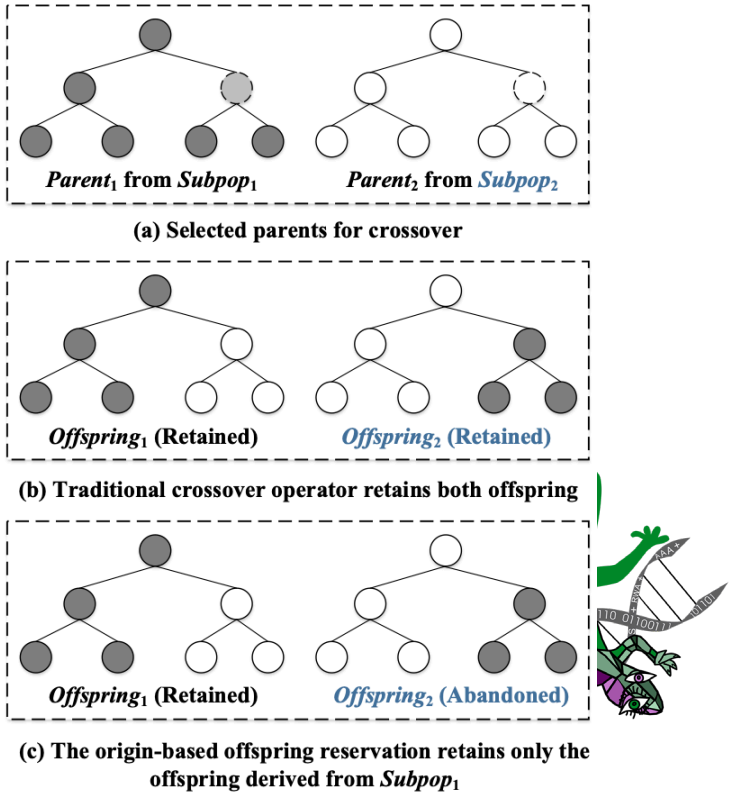
- Hardly possible to have a generalist heuristic for ALL possible scenarios
- Too much to learn specialist heuristic for each scenario
- Multitask Learning (parallel)
  - Learn heuristics for multiple scenarios simultaneously
- Transfer Learning (sequential)
  - Reuse knowledge to improve learning performance for new scenarios



Multitask GP

- Multiple subpopulations, each for a task
- subpopulations interact by crossover, retaining only one offspring more similar to the origin task

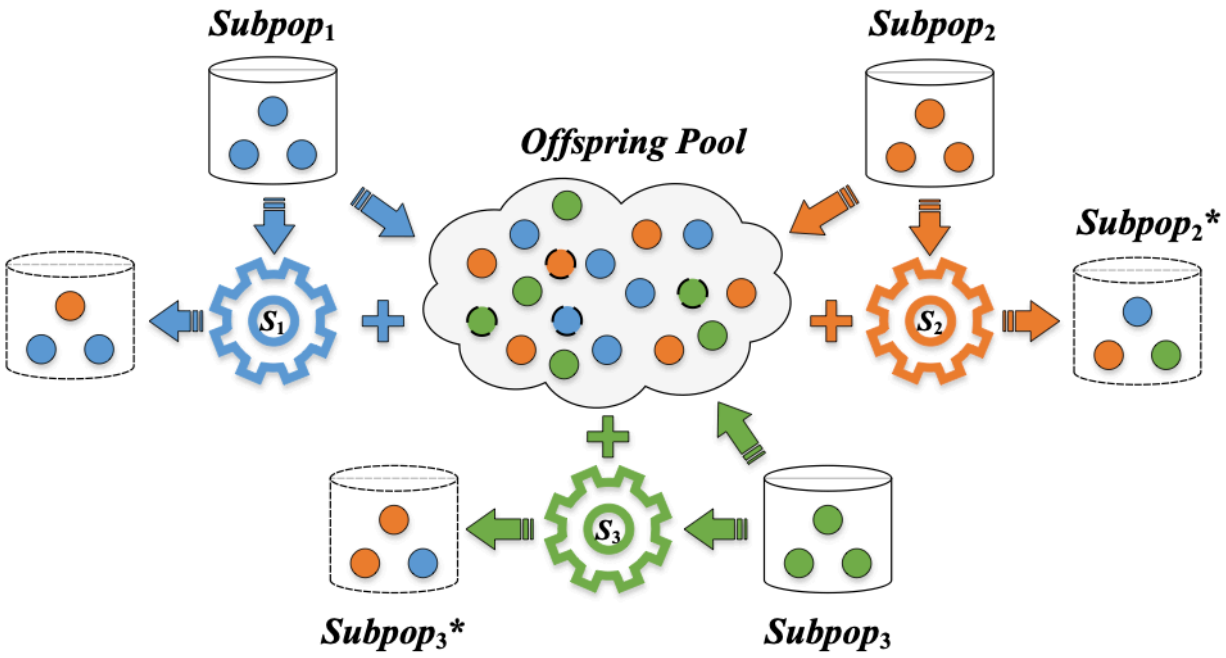


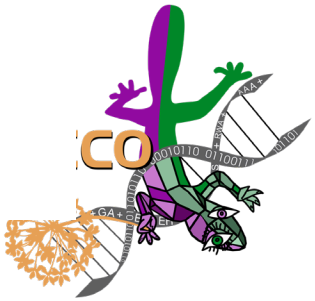


Zhang, F., Mei, Y., Nguyen, S., Tan, K. C., & Zhang, M. (2021). Multitask genetic programming-based generative hyperheuristics: A case study in dynamic scheduling. *IEEE Transactions on Cybernetics*, *52*(10), 10515-10528.

Multitask GP

Migrate offspring among sub-populations based on their effectiveness, evaluated by surrogate

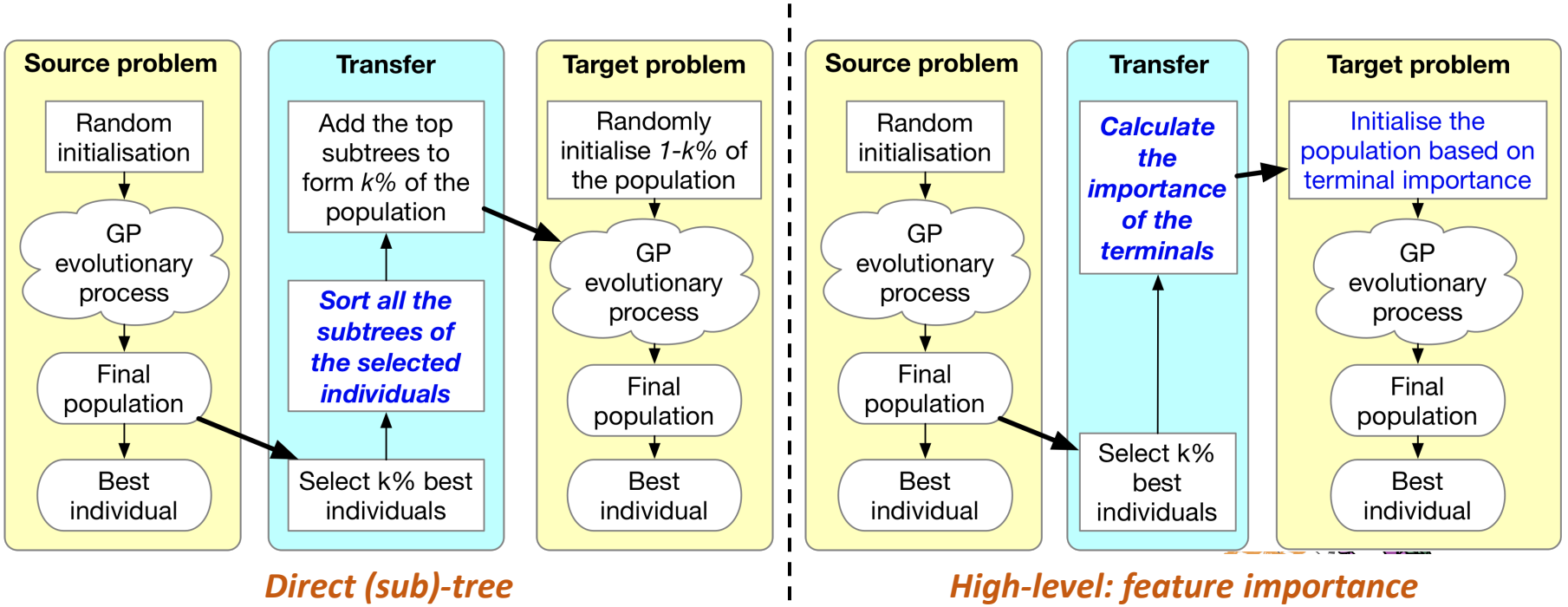




Zhang, F., Mei, Y., Nguyen, S., Zhang, M., & Tan, K. C. (2021). Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation*, *25*(4), 651-665.

GP with Knowledge Transfer

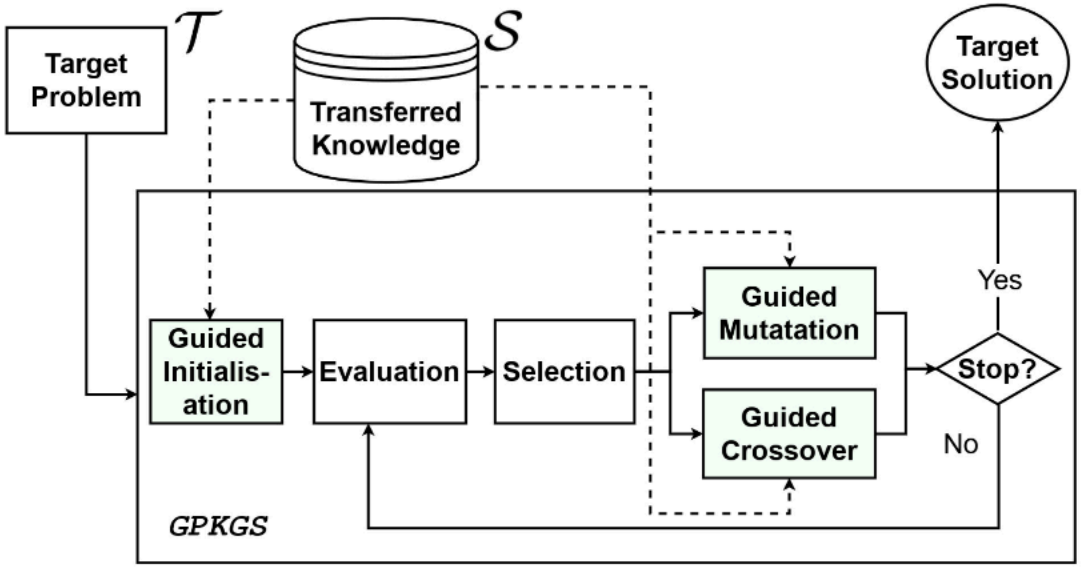
- What to transfer? Final (sub-)trees, feature importance, …
- When to transfer? Initialisation, breeding, …
- How to transfer? Directly transfer, adjust probabilities, …



Yi Mei, Mazhar Ansari Ardeh, and Mengjie Zhang. Knowledge transfer in genetic programming hyper-heuristics. In *Automated Design of Machine Learning and Search Algorithms*, pages 149--169. Springer, 2021

GP with Knowledge Transfer

- GP with Knowledge-Guided Search
  - Guided initialization: select the best inidividuals from the source domain
  - Guided crossover/mutation: to avoid individuals that have been seen in the source domain, to reduce negative transfer



Ardeh, M. A., Mei, Y., & Zhang, M. (2021). Genetic programming with knowledge transfer and guided search for uncertain capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation*, *26*(4), 765-779.

Machine Learning to Aid Meta-heuristics

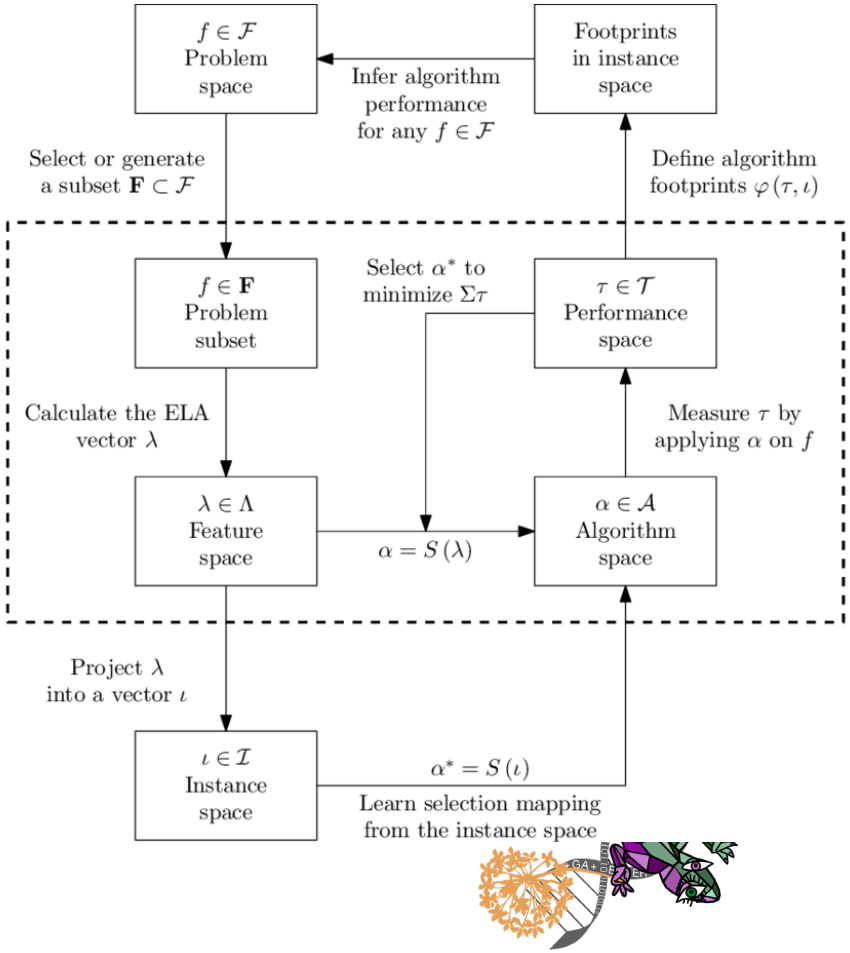
(including Evolutionary Methods)

ML to Aid Meta-heuristics

- Learn the design decisions in meta-heuristics
  - Algorithm selection: how to select the best algorithm for solving a given problem?
  - Algorithm composition: how to select the best components of the algorithm, such as initialization method, local search operators, genetic operators?
  - Algorithm configuration: how to automatically find the best parameter values?
- Offline vs Online
- Supervised vs Reinforcement Learning

Supervised Offline Algorithm Selection

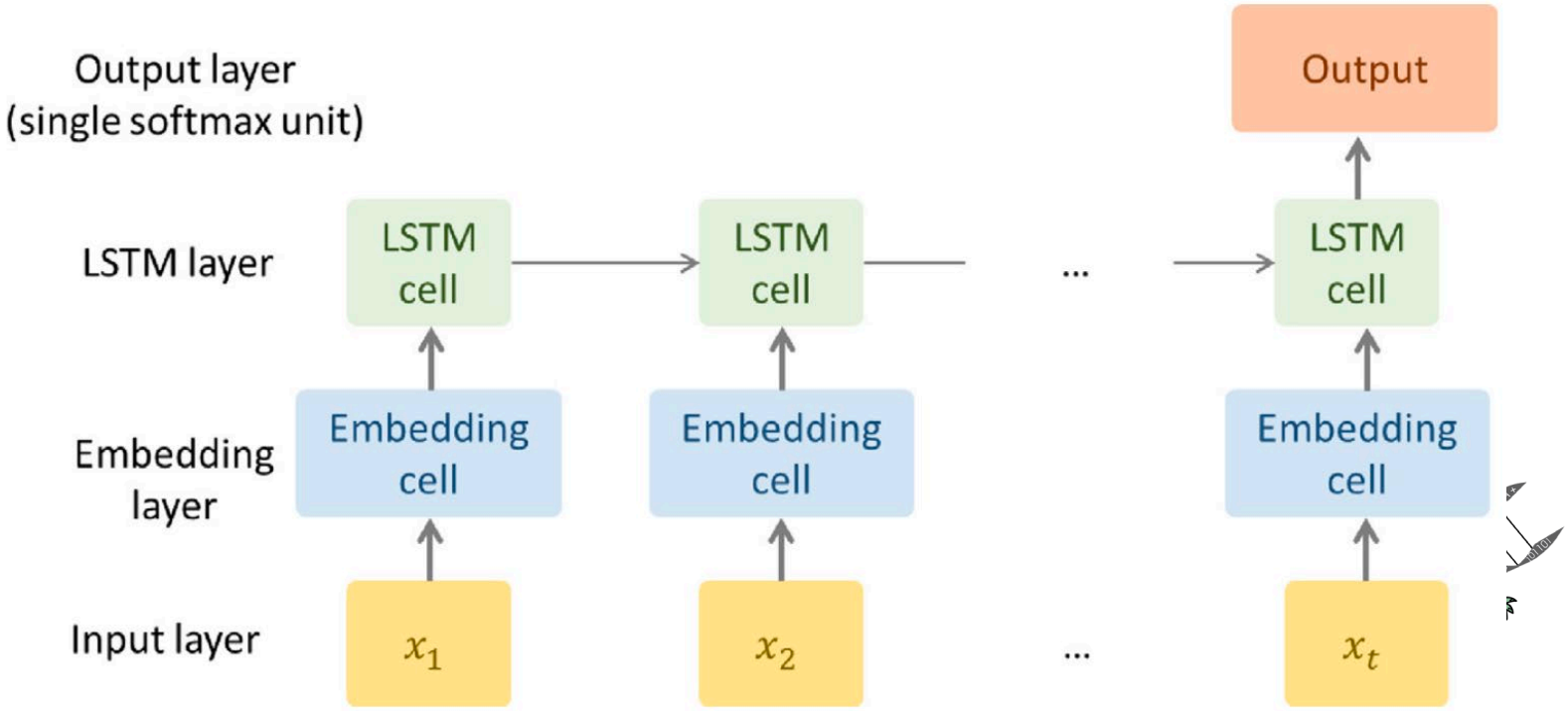
1. Collect instances and characterise them
2. Collect algorithms
3. Run algorithms to instances
4. Learn model from instance features to selected algorithms, or predicted algorithm performance



Smith-Miles, K., Baatar, D., Wreford, B., & Lewis, R. (2014). Towards objective measures of algorithm performance across instance space. *Computers & Operations Research*, *45*, 12-24.

Supervised Offline Algorithm Composition

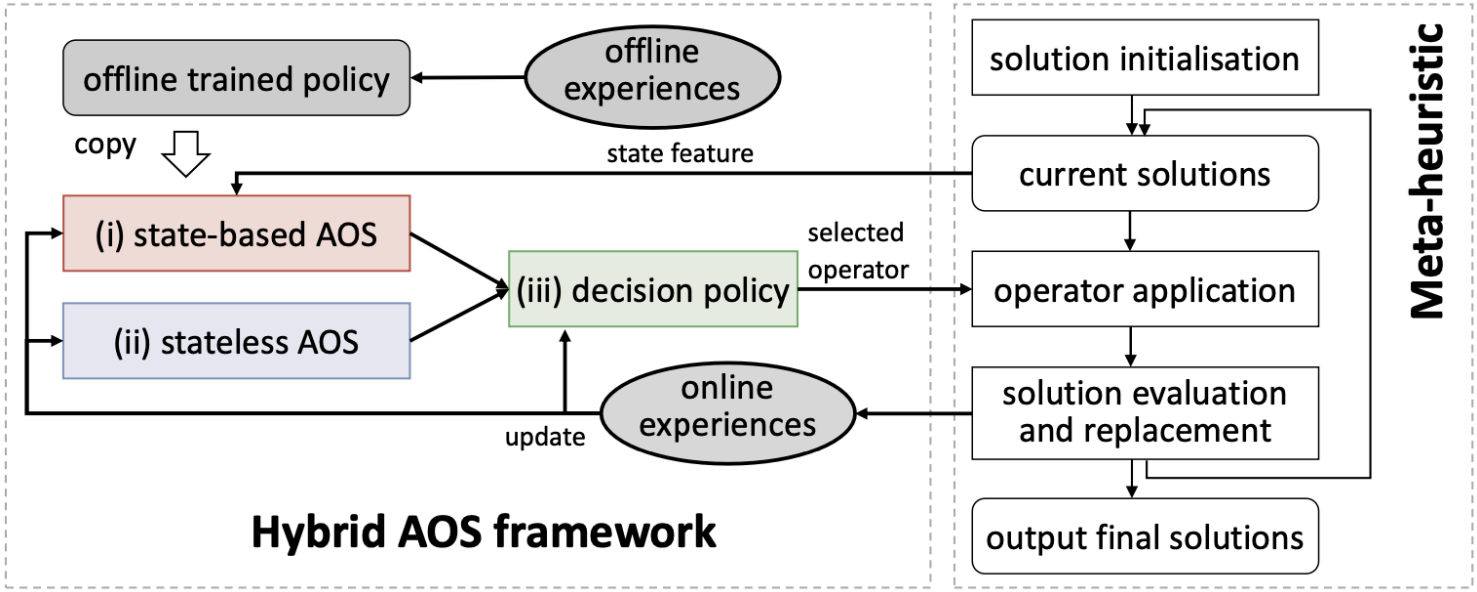
- Learning operator sequence through LSTM
  - Collect operator sequences
  - Learn LSTM for next operator prediction



Meng, W., & Qu, R. (2024). Automated design of local search algorithms: Predicting algorithmic components with LSTM. *Expert Systems with Applications*, *237*, 121431.

Reinforcement Algorithm Composition

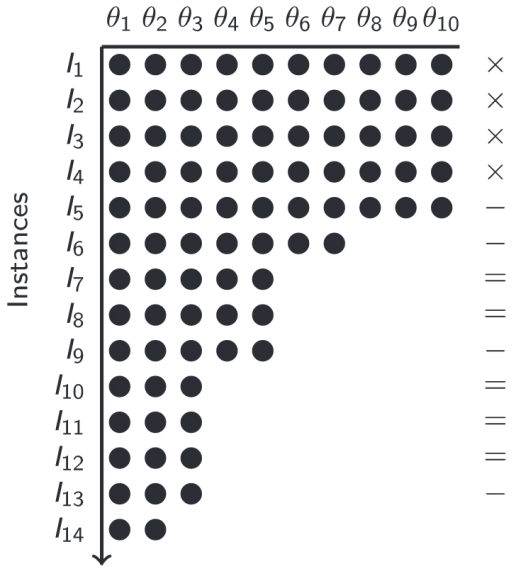
- Offline + Online Adaptive Operator Selection
  - Train state-based policy (when to select which operator) offline from training data
  - Update policy online through both state-based and stateless policy for new problem instance

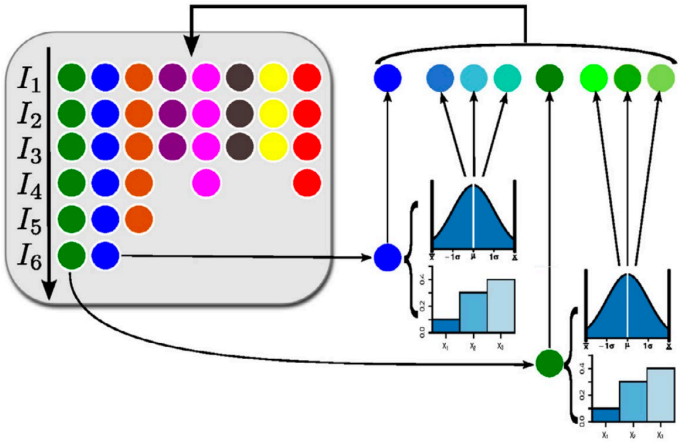


Jiyuan Pei, Jialin Liu, and Yi Mei. Learning from offline and online experiences: A hybrid adaptive operator selection framework. In *Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO)*. ACM, jul 2024

Reinforcement Algorithm Configuration

- Irace: iterative racing for automatic parameter tuning
- Repeat:
  - sampling new configurations according to a particular distribution,
  - selecting the best configurations from the newly sampled ones by means of racing,
  - updating the sampling distribution in order to bias the sampling towards the best configurations.



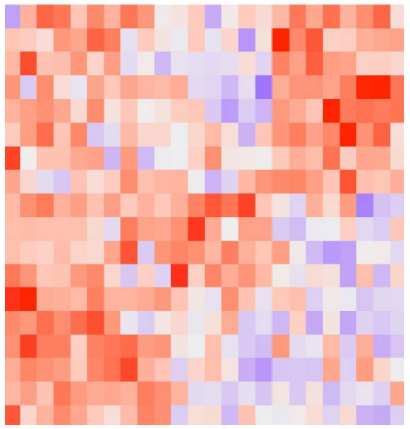


López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, *3*, 43-58.

Learning Heatmaps

Learn model indicating likelihood for

- components, e.g., edges in TSP, VRP, nodes in MISP, MaxCut…
- to appear in (close to) optimal solutions



Can be used to

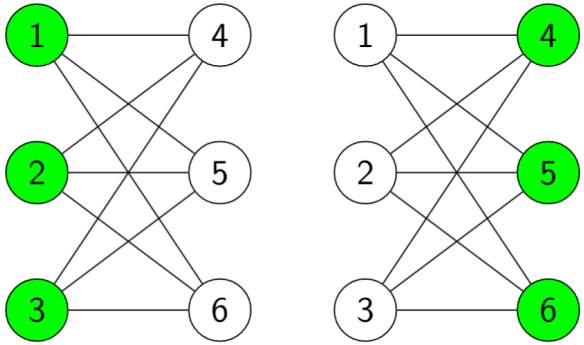
- prune/sparsen the search space
- bias or reduce operators towards inclusion of more promising components

Training

- supervised based on (close to) optimally solved, representative instances
- reinforcement learning

Issues of Heatmaps: 1. Unimodality

Example:
Maximum independent set problem:

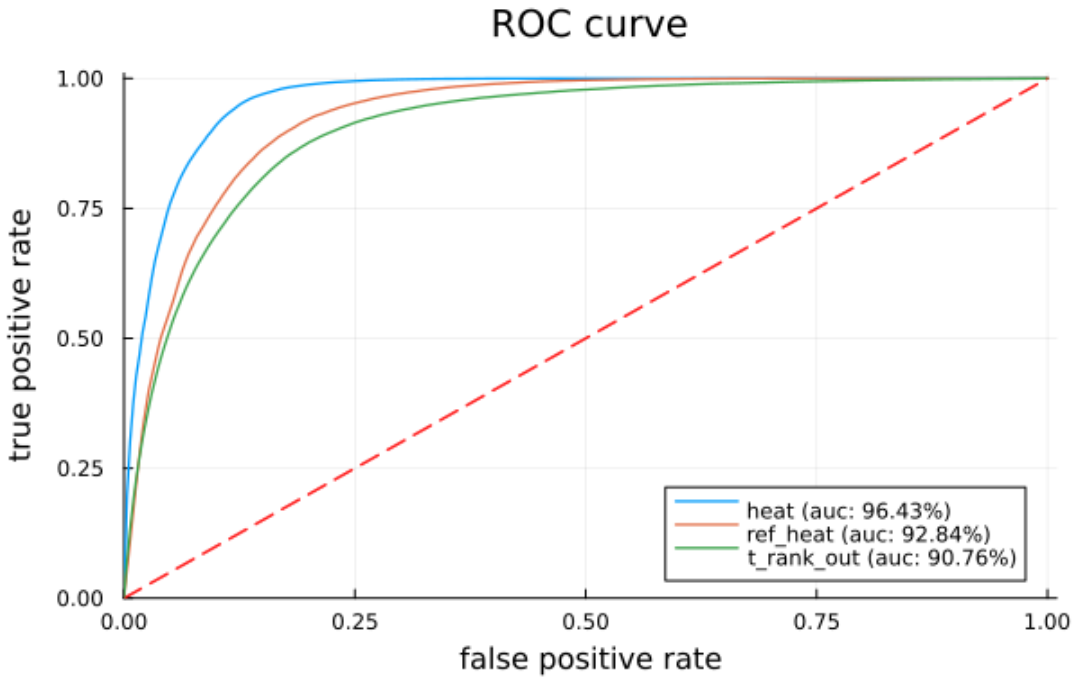


Heatmap: all nodes equally likely in an optimal solution
-> no meaningful information

Symmetries and different (close to) optimal solutions often cause problems.

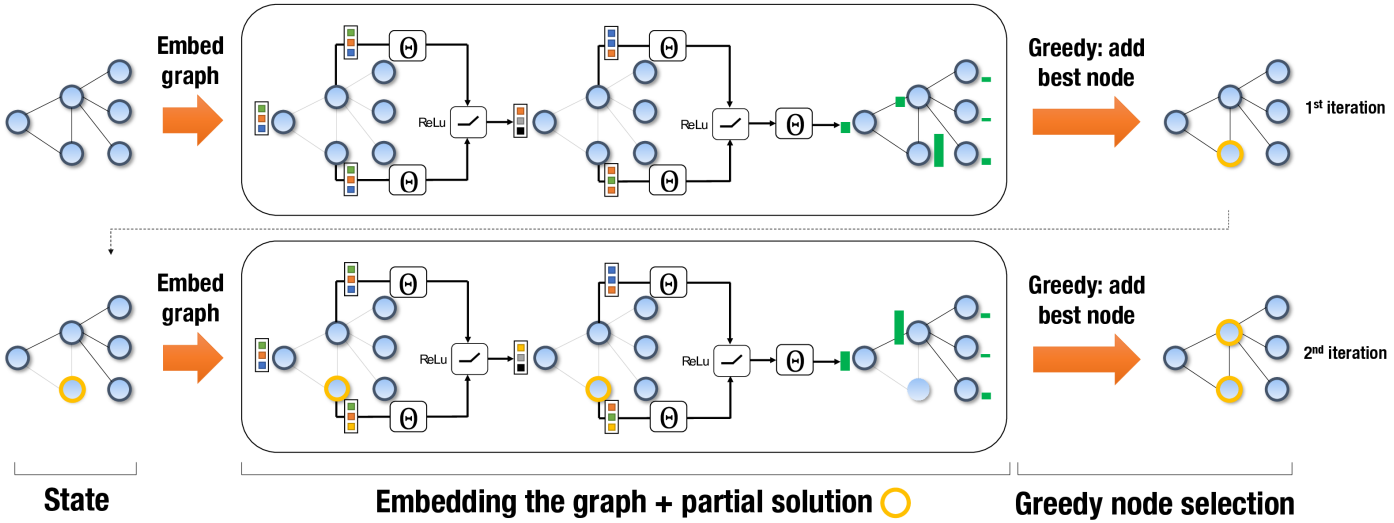
Issues of Heatmaps:

2. Some essential components often do not have characteristics of typical solution components



M. Bresich et al. (2025): Search Space Reduction Through MachineLearning for the Electric Autonomous Dial-A-Ride Problem, submitted

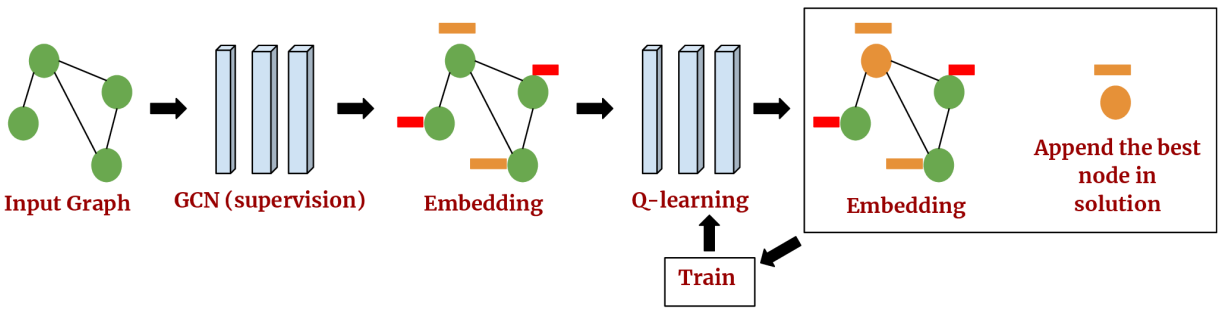
Learning to Solve Graph Problems



- H. Dai, E.B. Khalil, Y. Zhang, B. Dilkina, and L. Song (2017): Learning combinatorial optimization algorithms over graphs. NeurIPS
- Min vertex cover, max cut, TSP considered
- Graph neural network (GNN) used, structure2vec used to "featurize" nodes
- variant of Q-learning used to obtain a policy for constructing solutions in an auto-regressive way

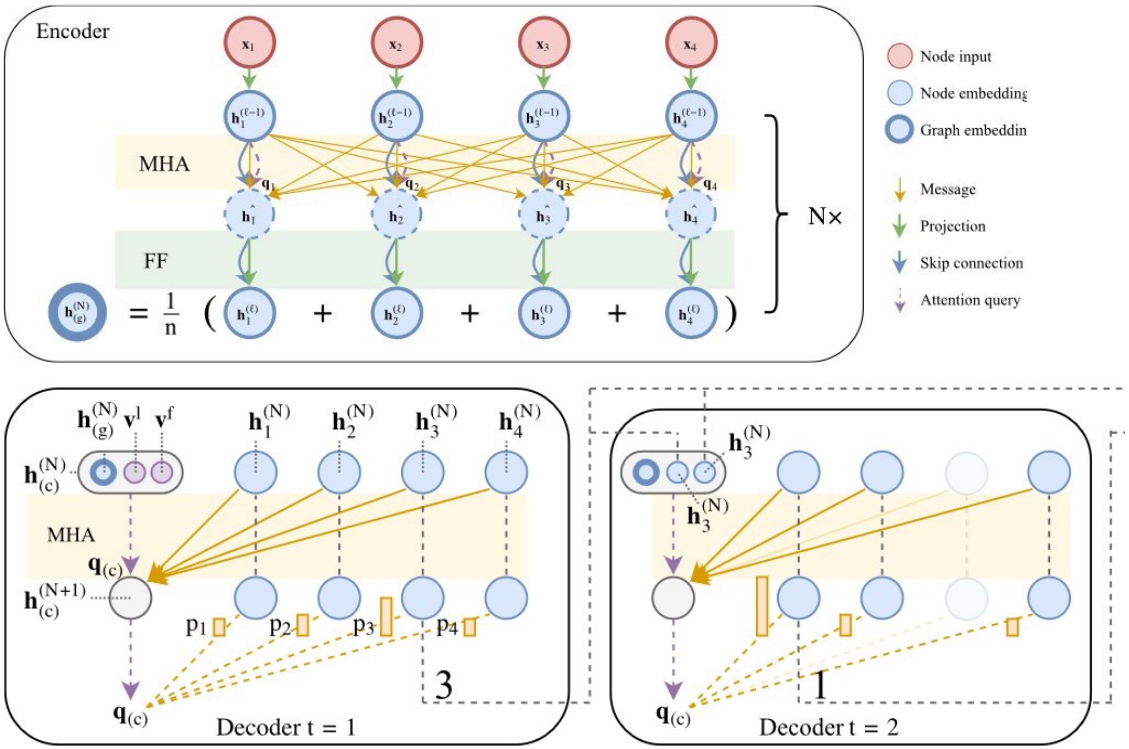
Learning to Solve Graph Problems (cont.)

- A. Mittal, A. Dhawan, S. Manchanda, S. Medya, S. Ranu, A. Singh (2020): Learning Heuristics over Large Graphs via Deep Reinforcement Learning. NeurIPS.
- focus on scalability to larger graphs, for max influence, max vertex cover, max coverage problems
- GCN trained in supervised fashion to predict likelihood of each node to be part of a solution
- Q-learning finds combination of most promising nodes $Q(S, v)$: long-term reward for adding node v to solution S approximated by a simpler GNN



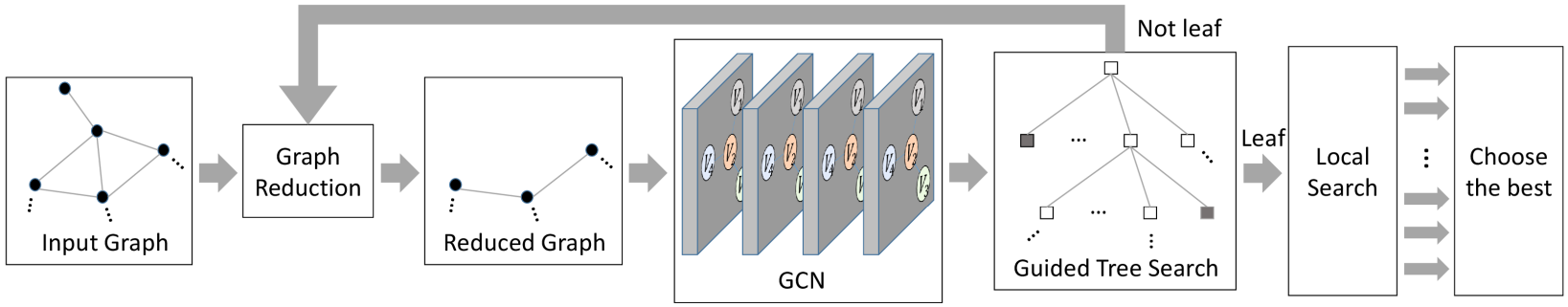
Learning to Solve Graph Problems (cont.)

- W. Kool, H. van Hoof, M. Welling (2019): Attention, learn to solve routing problems!, ICLR
- Autoregressive attention-based encoder + decoder GNN
- for TSP, VRP; trained with REINFORCE



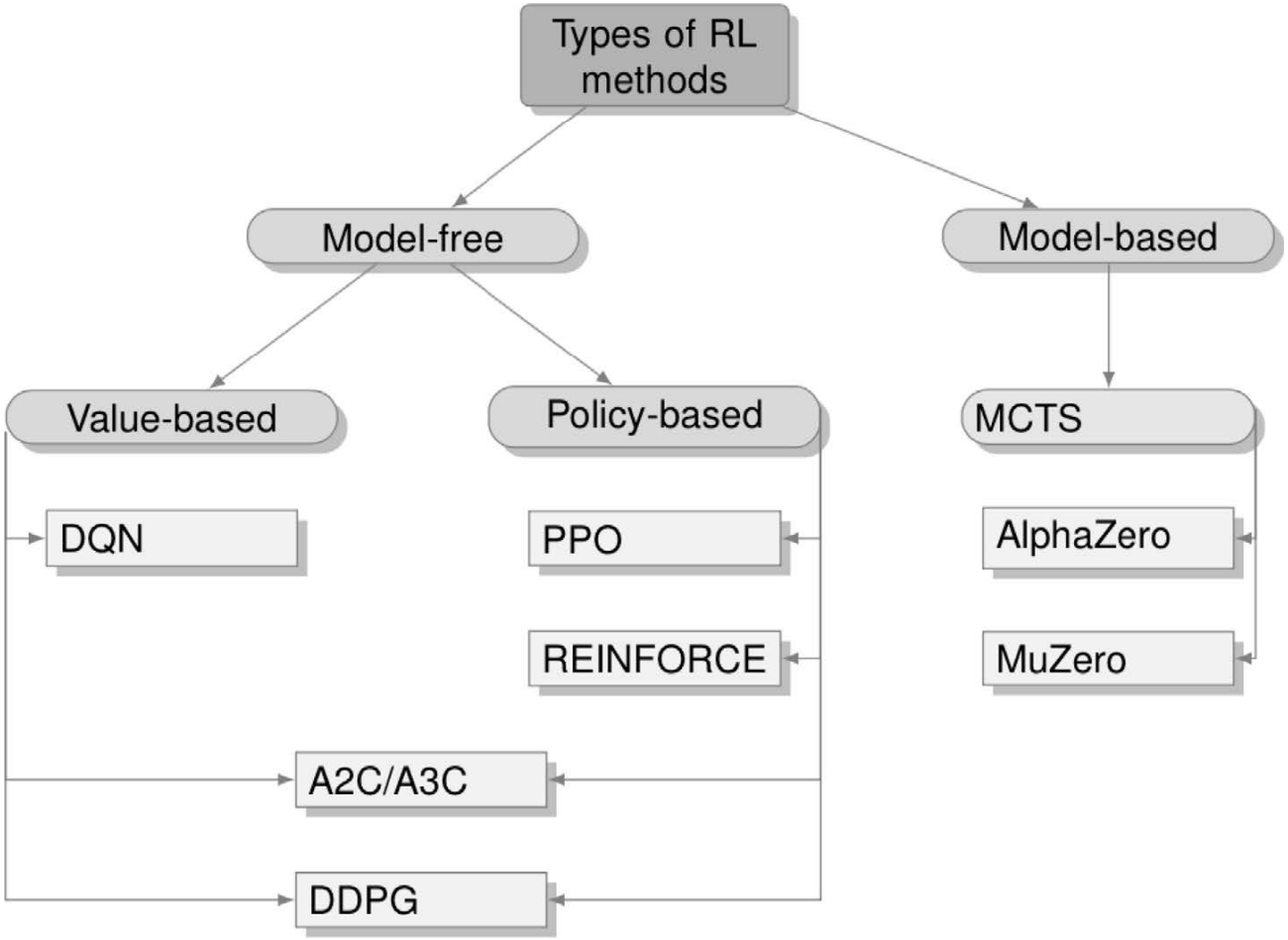
Learning to Solve Graph Problems (cont.)

- Z. Li, Q. Chen, V. Koltun (2018): Combinatorial optimization with graph convolutional networks and guided tree search, NeurIPS



- for max independent set, min vertex cover, max clique, SAT
- GNN used to predict likelihood of each node to be part of a solution; yields multiple probability
- heuristic tree search utilizing multiple maps, graph reduction, local search applied
- supervised learning

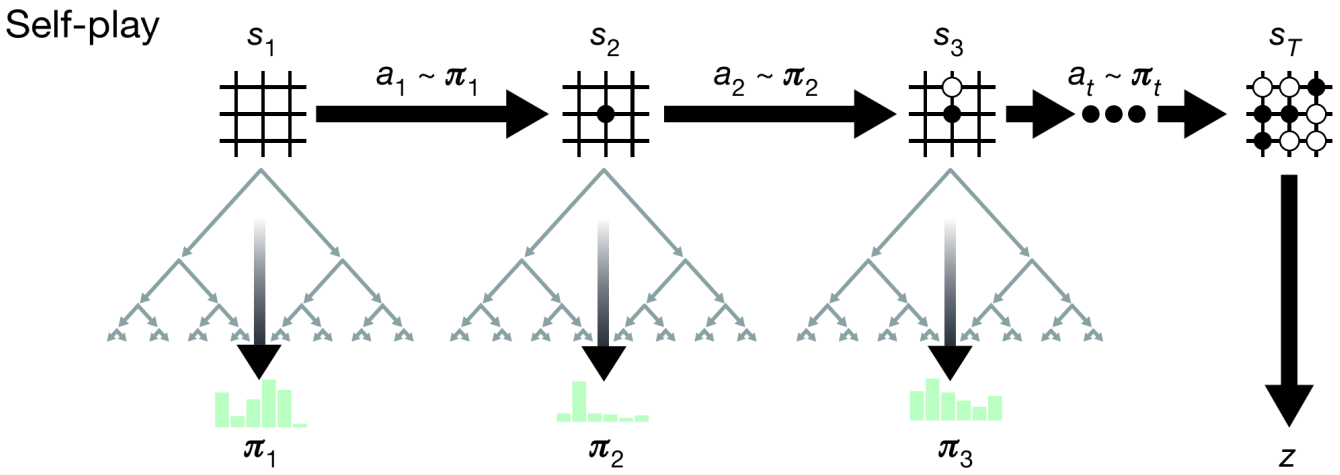
Reinforcement Learning (RL) - Classification



from Mazyavkina et al. (2021): Reinforcement learning for combinatorial optimization: A Survey, C&OR 134(105400)

Basic Idea of AlphaZero

- D. Silver et al. (2018). A general reinforcement learning algorithm thatmasters chess, shogi, and Go through self-play. Science, 362(6419)
- Learns only by iterated selfplay
- Monte Carlo Tree Search (MCTS) applied to obtain a policy and select a move
- States are evaluated by a Deep NN
  Quality of NN output is amplified by MCTS!

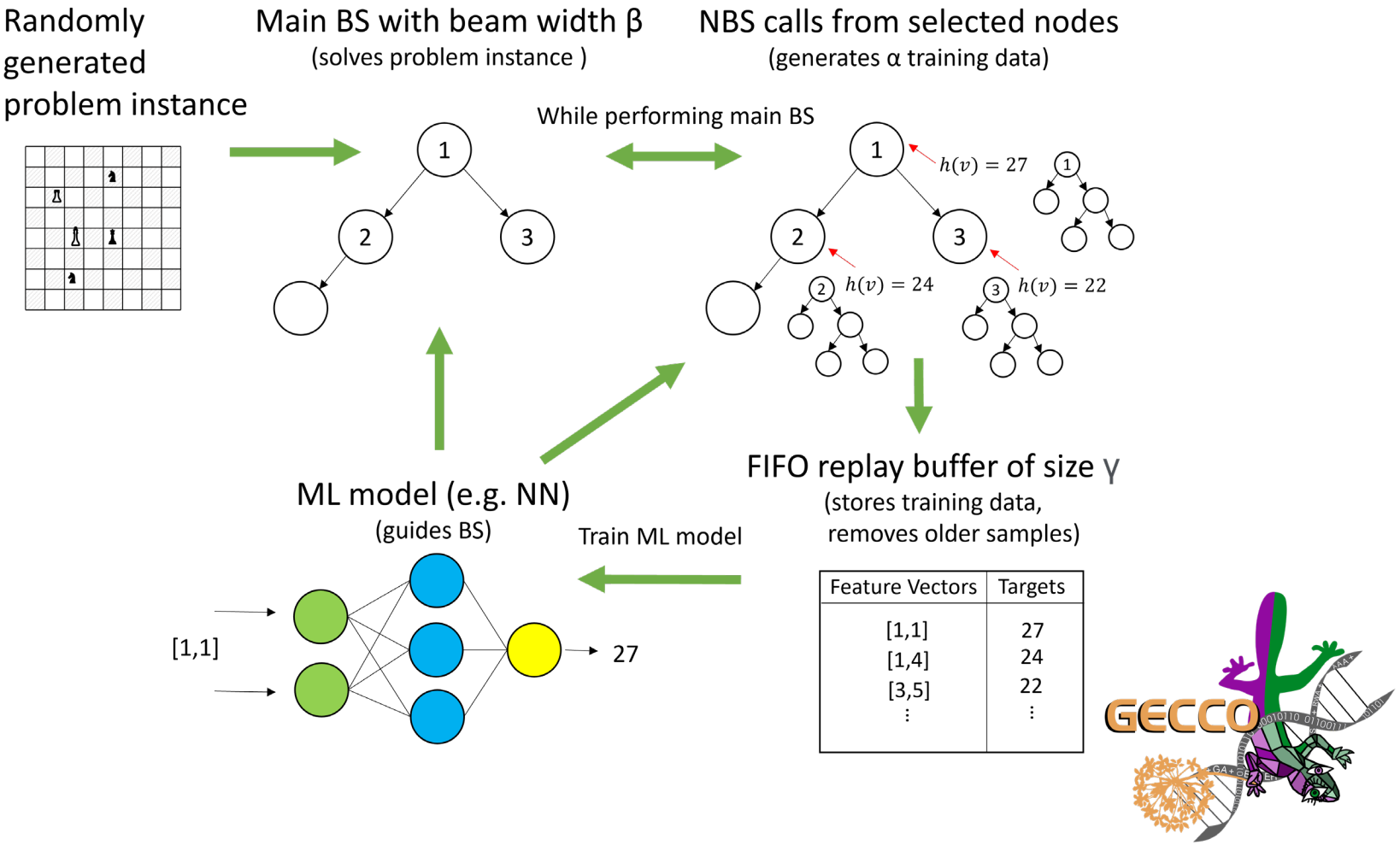


Learning to Solve Graph Problems (cont.)

- K. Abe, Z. Xu, I. Sato, M. Sugiyama, M. (2020): Solving NP-hard problems on graphs with extended AlphaGo zero.arXiv:1905.11623
- for min vertex cover, max cut, max clique problems
- based on AlphaGoZero
- different GNNs tested
- special reward normalization applied

- J. Huang, M. Patwary, and G. Diamos (2019): Coloring big graphs with AlphaGoZero. arXiv:1902.10162
- similar approach for coloring huge graphs
- special FastColorNet neural network architecture

Learning Beam Search (LBS)



M. Huber and G. Raidl (2021). Learning beam search: Utilizing machine learning to guide beam search for solving combinatorial optimization problems. Machine Learning, Optimization, and Data Science (LOD 2021)

LBS for the 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.

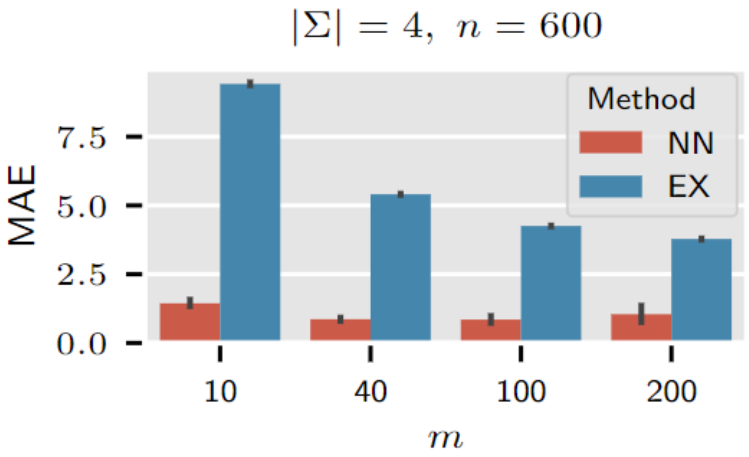
  Example: $m = 2$, $|\Sigma| = 3$

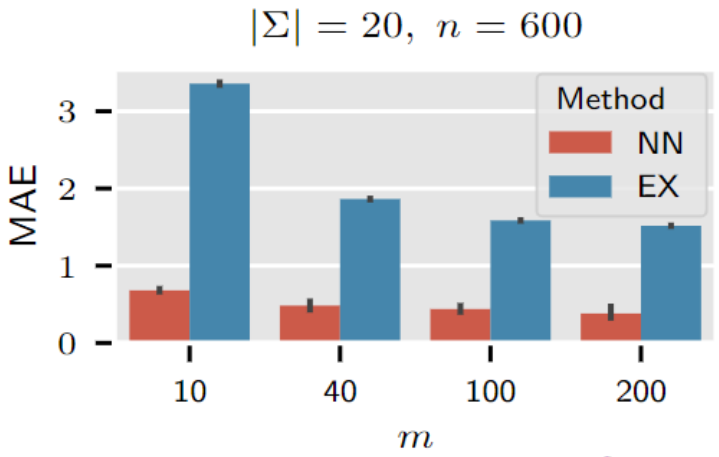
$$\begin{matrix} s_1\text{: ABBA} \\ s_2\text{: CABA} \end{matrix} \Rightarrow \text{ABA}.$$

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

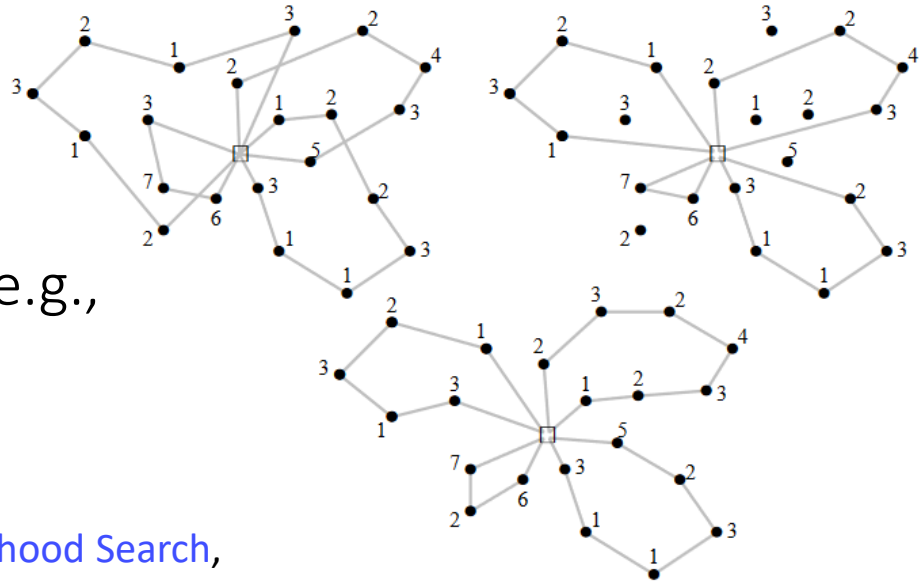
- Results on rat and BB LCS benchmark instances:
  - NN: MLP with 20+20 hidden nodes
  - Features: remaining input string lengths, remaining min. letter occurrences
  - Beam width:
    - LBS training done with $\beta = 50$
    - Low computation time tests with $\beta = 50$ on 13 out of 28 instances new best results
    - High quality tests with $\beta = 600$ on 7 out of 28 instances new best results
- Also successfully considered: Constrained LCS, shortest common supersequence problem, no-wait flow shop problem

Learning in Large Neighborhood Search

Main principle of Large Neighborhood Search (LNS):

- Iteratively apply destroy and repair operators
  - Destroy: e.g., remove orders from a VRP tour,
    remove values assigned to variables
  - Repair: e.g., fast greedy heuristics, dynamic progarmming,
    solve a mixed integer linear program

- One of the most prominent metaheuristics, e.g., in transport optimization



D. Pisinger, S. Ropke (2010): Large Neighborhood Search,
Handbook of Metaheuristics, 2nd edition, 399-420, Springer

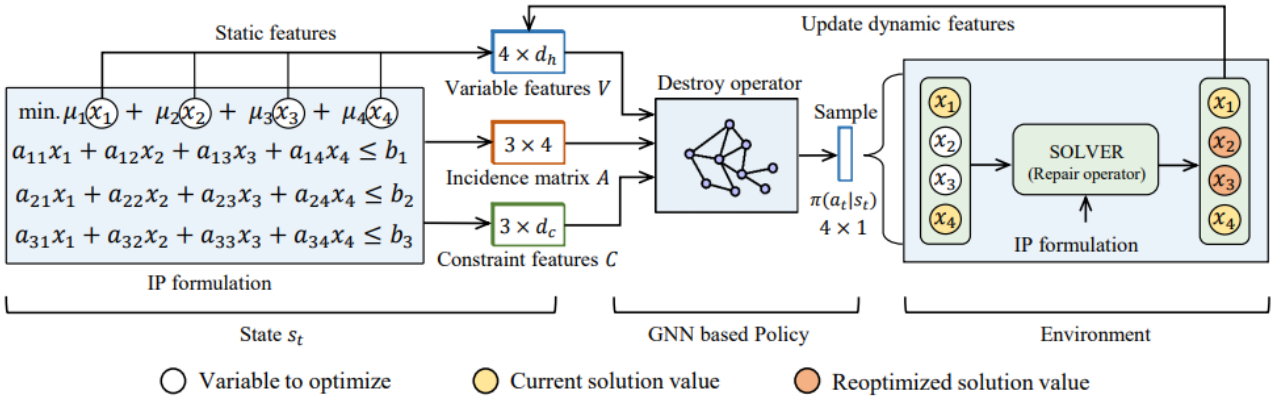
Learning in LNS:

Selection of destroy/repair operators, e.g.,

- J. Kallestad, R. Hasibi, A. Hemmati, K. Sörensen (2023): A General Deep Reinforcement Learning Hyperheuristic Framework for Solving Combinatorial Optimization Problems, EJOR
- S.-N. Johnn, V.-A. Darvariu, J. Handl, J. Kalcsics (2023): Graph Reinforcement Learning for Operator Selection in the ALNS Metaheuristic, OLA.

Selection of what to destroy, e.g.,

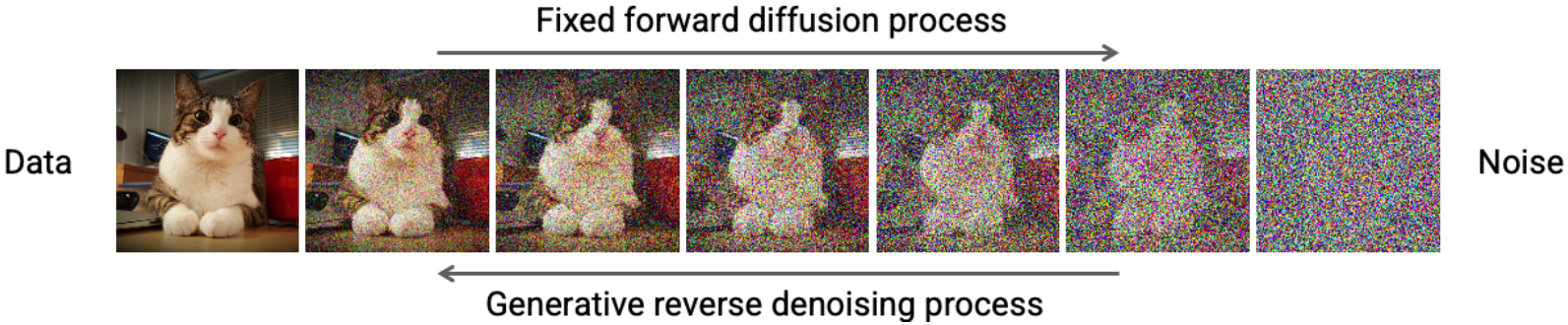
- J. Song, R. Lanka, Y. Yue, B. Dilkina (2020): A General Large Neighborhood Search Framework for Solving Integer Linear Programs, NeurIPS
- R. Addanki, V. Nair, M. Alizadeh (2020): Neural Large Neighborhood Search, NeurIPS
- N. Sonnerat, P. Wang, I. Ktena, S. Bartunov, V. Nair (2021): Learning a Large Neighborhood Search Algorithm for Mixed Integer Programs, arXiv:2107.10201
- Y. Wu, W. Song, Z. Cao, J. Zhang (2021): Learning Large Neighborhood Search Policy for Integer Programming, NeurIPS



- F. Oberweger, G. Raidl, E. Rönnberg, and M. Huber (2022): A Learning Large Neighborhood Search for the Staff Rerostering Problem, CPAIOR
- T. Huang, A. Ferber, Y. Tian, B. Dilkina, B. Steiner (2023): Searching Large Neighborhoods for Integer Linear Programs with Contrastive Learning, ICML

Denoising Diffusion Models (DDMs)

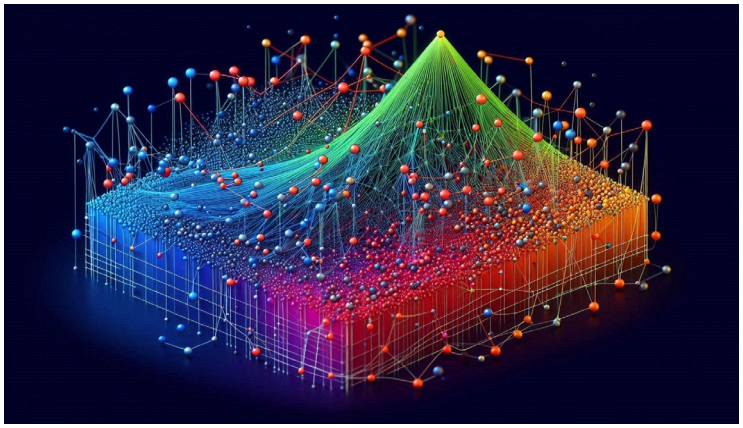
- State-of-the-art in many generative AI applications, in particular the creation of realistically-looking images



- 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 Comb. Opt.

(Sun and Yang, NeurIPS 2023)



- TSP and MISP considered
- utilizes an anisotropic GNN with edge gating
- discrete diffusion based on Bernoulli noise
- trained on many small instances + (close to) optimal solutions
- (Possible) advantages
  - outperforms earlier approaches by a large margin in their tests
  - faster than autoregressive models
  - better scaling to larger instances
  - **multi-modality** of solution space is considered

DIFUSCO: Graph-Based Diffusion Solver for Comb. Opt.

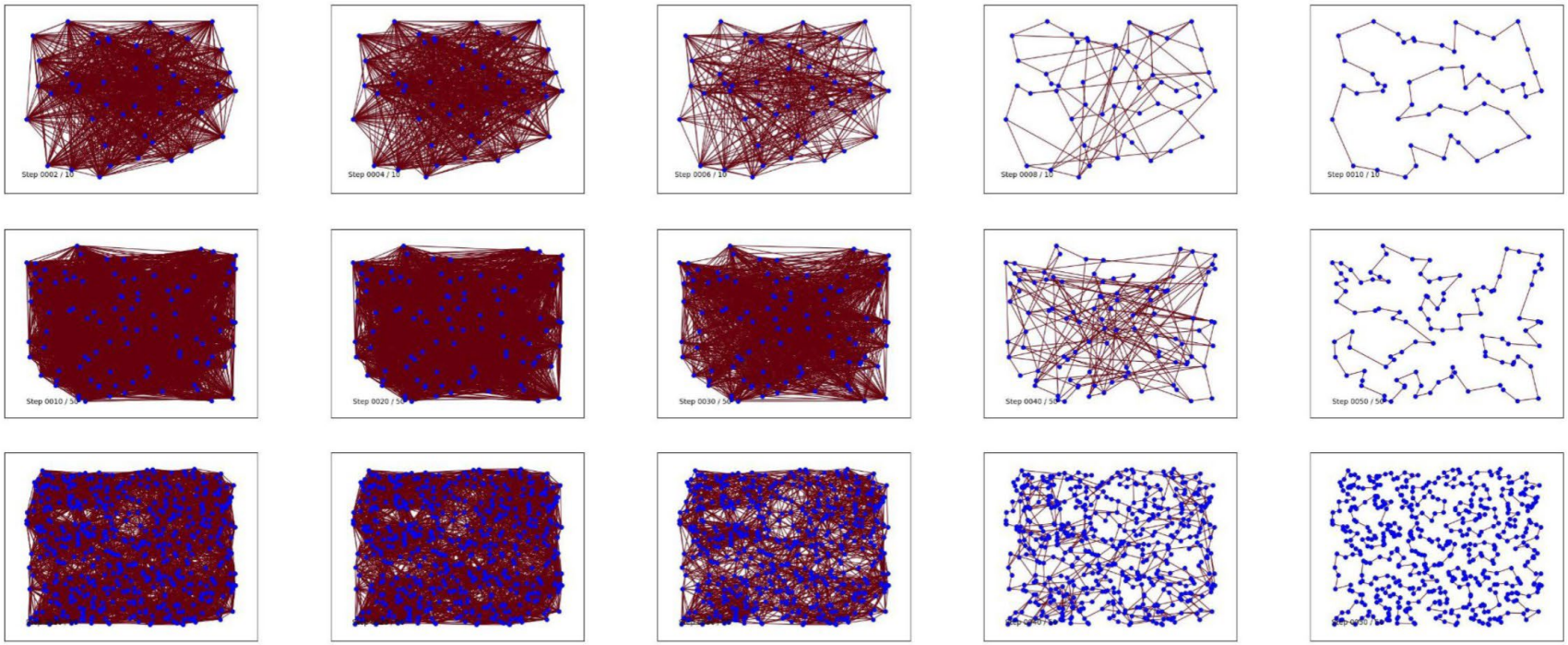


Figure 11: Qualitative illustration of discrete DIFUSCO on TSP-50, TSP-100 and TSP-500 with 50 diffusion steps and `cosine` schedule.

(from Sun and Yang, 2023)

DIFUSCO: Graph-Based Diffusion Solver for Comb. Opt.

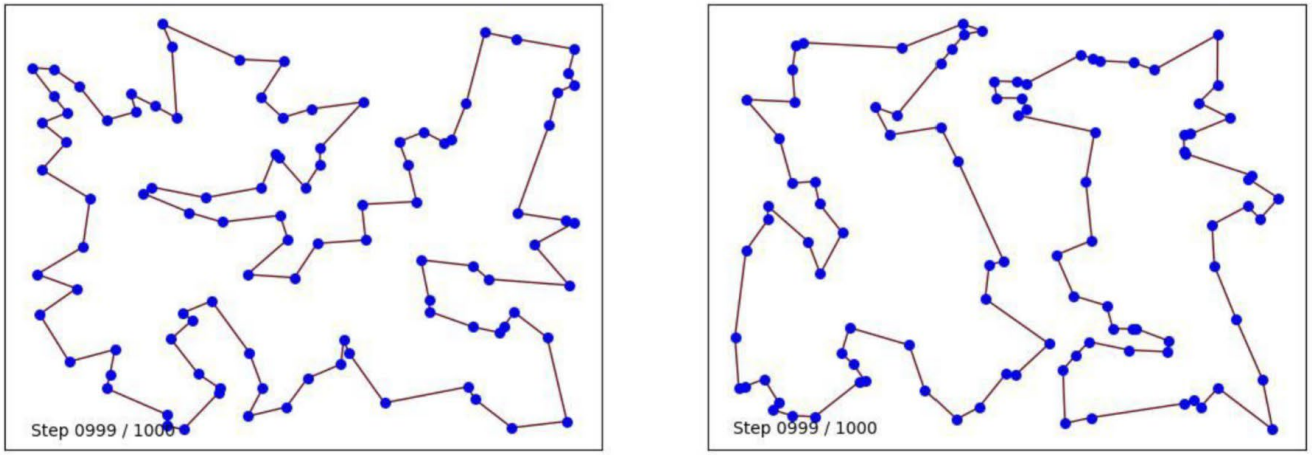
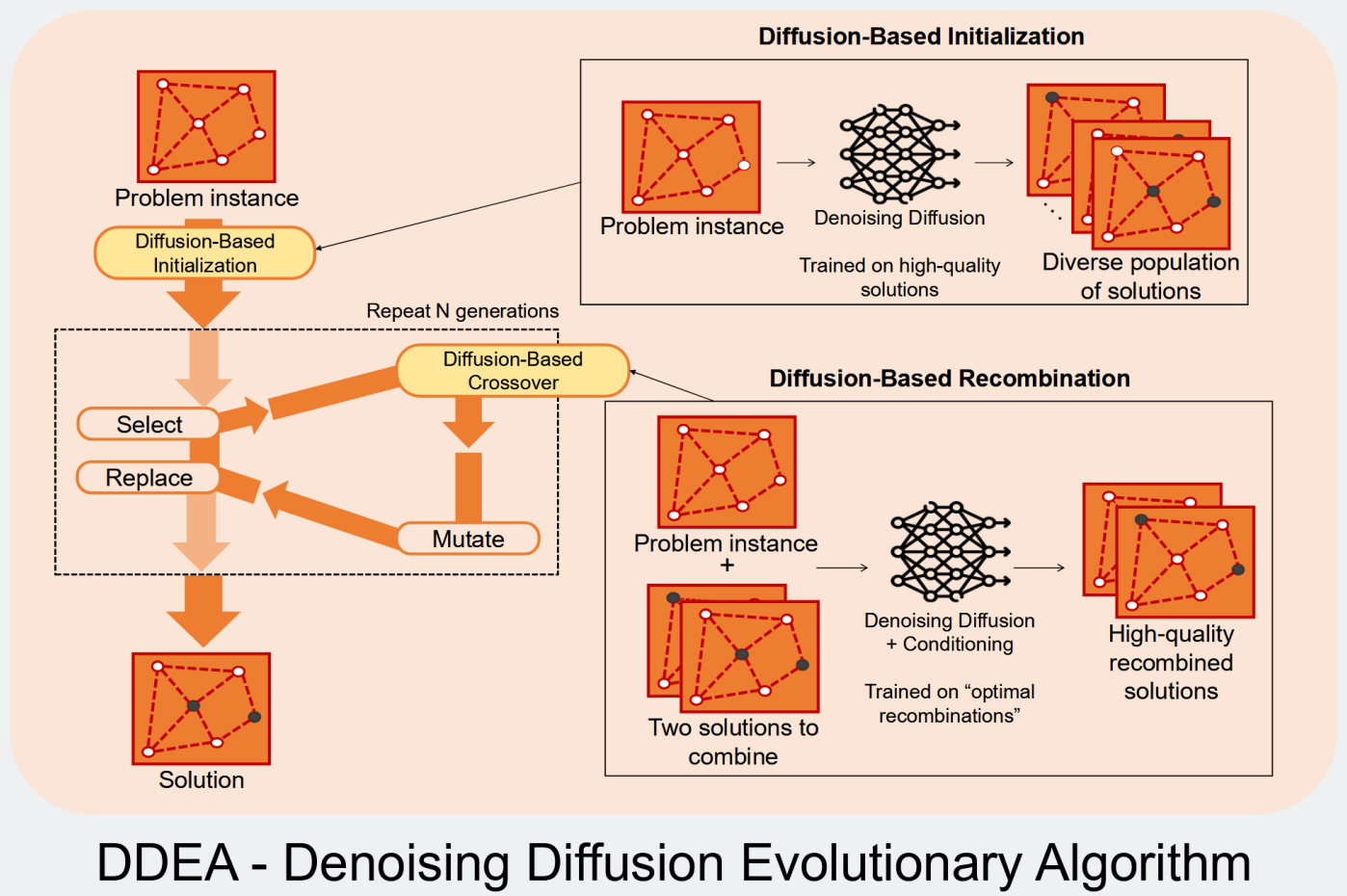


Figure 12: Success (left) and failure (right) examples on TSP-100, where the latter fails to form a single tour that visits each node exactly once. The results are reported without any post-processing.

(from Sun and Yang, 2023)

Denoising Diffusion Based Evolutionary Algorithm

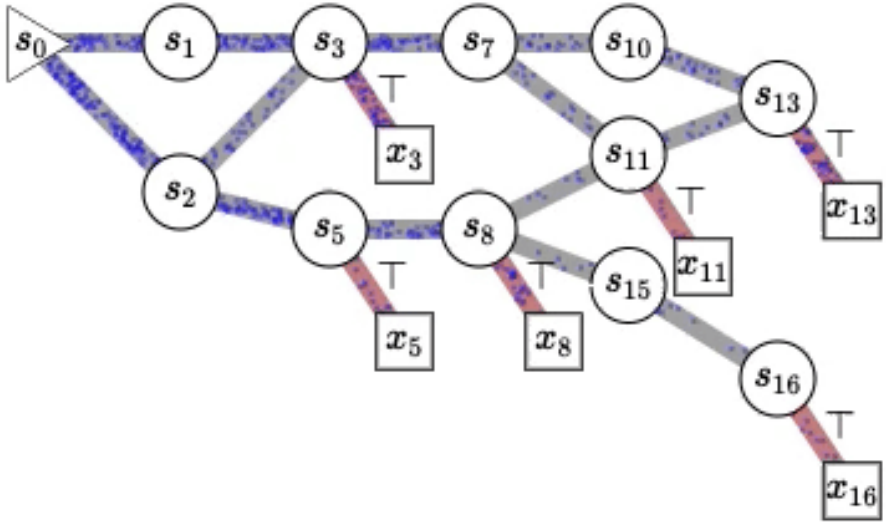


J. Salva Soler and G. R. Raidl (2025): A Denoising Diffusion-Based Evolutionary Algorithm Framework Demonstrated on the Maximum Independent Set Problem, LOD 2025

Generative Flow Networks (GFlowNets)

(Bengio, 2021)

- Stochastic policy
- to construct objects through a sequence of steps
- with probability proportional to a reward function



(image from https://yoshuabengio.org/2022/03/05/generative-flow-networks)

- Related to, but different from classical RL

Generative Flow Networks (GFlowNets) – contd.

Advantages

- Emphasis on multi-modality
- Possibly more robust training than classical RL
  - Can be trained off-policy/offline
  - Does not necessarily rely on intermediate rewards
  - Models symmetries more efficiently

- About 60 papers published on GFlowNets since 2021

- Ongoing work: utilizing GFlowNets for
  - destroy set generation in LNS
  - as hyper-heuristic to schedule lower-level heuristics

Challenges and Future Directions

Challenges and Future Directions

Highly active research area, learning promising to improve on classical optimization in combinatorial optimization, BUT:

- End-to-end ML approaches are usually not (yet) competitive to carefully crafted classical approaches for COPs
  - Most often, combinations appear most promising
- Generalization to larger instances or instances of other types often is hard
- Obtaining training data and training often very time-consuming
- Deep reinforcement learning does not need labeled data, but often is tricky in practice
- Understand/Interpret the learned algorithms