# **Evolutionary Computation meets Machine Learning for Combinatorial Optimisation**

## **GECCO 2024 Tutorial**

Yi Mei, Associate Professor, Victoria University of Wellington, [yi.mei@ecs.vuw.ac.nz](mailto:yi.mei@ecs.vuw.ac.nz)

Guenther Raidl, Professor, TU Wien, [raidl@ac.tuwien.ac.at](mailto:raidl@ac.tuwien.ac.at)

### **Description**

*(Description of the tutorial scope and content; this field will also be displayed on the website as the default description for your tutorial if it is accepted.)*

Combinatorial optimisation is an important research area with many real-world applications such as scheduling, vehicle routing, cloud resource allocation, supply chain management, logistics and transport. Most combinatorial optimisation problems are NP-hard, making it challenging to design effective algorithms to solve them to optimality. In practice, in particular (meta-)heuristic methods including evolutionary approaches are therefore widely used to address such problems. Unfortunately, designing an effective and efficient (meta-)heuristic typically requires extensive domain expertise and a lot of trial and error for each different problem variant encountered in the real world.

In recent years, machine learning has emerged to also be a promising ingredient for better and/or easier solving combinatorial optimisation problems. First, machine learning can design combinatorial optimisation algorithms *automatically* by searching for algorithms/heuristics rather than solutions, and the learned algorithms/heuristics can be generalised to future unseen problem variants to obtain high-quality solutions. This can greatly reduce the dependence on human expertise and time to manually design effective algorithms. Second, machine learning can learn decision-making policies for dynamic combinatorial optimisation problems (e.g., dispatching rules for dynamic scheduling), which can achieve both effectiveness and efficiency simultaneously. Third, machine learning may discover new design patterns and knowledge that can further improve the algorithm design for solving complex combinatorial optimisation problems.

The aim of this tutorial is two-fold. On the one hand, we will give an overview on how classical metaheuristics may profit from the usage of machine learning and provide a few advanced examples. On the other hand, we will introduce how evolutionary machine learning, specifically, can be used for solving combinatorial optimisation problems, including basic design issues and some case studies.

The outline of this 110-minute tutorial is as follows.

1. Introduction and Background [20 mins]
   1. Combinatorial Optimisation: Problems and Methods
   2. Machine Learning
   3. Metaheuristics including Evolutionary Computation
   4. Examples to show diverse classical metaheuristics already include (simple) learning mechanisms since decades
   5. Basic taxonomy of metaheuristics involving more rigorous learning
   6. Machine Learning for Combinatorial Optimisation *(borrow from tutorials about ML4CO)*
      1. Learning to Construct Solutions
      2. Learning to Search in the Solution Space
2. Evolutionary Computation to Learn Combinatorial Optimisation Heuristics [40 mins]
   1. Overall Framework
   2. Basic Design Issues
      1. Individual Encoding (Search Space Definition)
      2. Fitness Function (Training Dataset, Performance Metrics, …)
      3. Search Process (Genetic Operators, Speedup by Surrogate, …)
      4. Generalisation
      5. Knowledge Transfer and Multitask Optimisation
      6. Interpretability
   3. Case Studies
      1. GP to learn variable ordering heuristics in constraint programming
      2. GP to learn dispatching rules for dynamic scheduling
      3. GP to learn ambulance dispatching policies (with demo)
      4. …
3. Machine Learning to Learn Metaheuristics [40 mins]
   1. ML-assisted EC algorithms for combinatorial optimisation
      1. Automatic algorithm configuration/parameter tuning (IRACE)
      2. Adaptive operator selection
      3. Grammar-guided GP to design EAs
   2. Learning to solve graph problems with graph neural networks
   3. Learning to design large neighbourhood search
      1. A case study on staff re-rostering problem
   4. Learning to search
      1. Learning beam search
      2. AlphaZero-like approaches
4. Challenges and Future Directions [10 mins]

### **Expected Number of Participants**

100

### **Potential Audience**

*(A description of who might be interested in the tutorial.)*

The following groups of audience might be interested in the tutorial.

* **Researchers in combinatorial optimisation**: they will find new emerging pathways for designing effective combinatorial optimisation algorithms more easily.
* **Researchers in evolutionary computation**: they will find different research directions to apply evolutionary computation to complex real-world (combinatorial) optimisation problems and get ideas on how to utilize machine learning in metaheuristics.
* **Researchers in machine learning**: they will find a non-mainstream population-based gradient-free machine learning paradigm that can effectively applied to solve complex real-world problems and see how machine learning can possibly be utilized within classical metaheuristics.
* **Real-world practitioners**: they will learn new concepts to solve their practical problems possibly better or more easily (e.g., scheduling, management, decision-making under uncertainty).

### **Activities**

*([Highly encouraged] A description of any interactive activity or demo planned within the tutorial presentation.)*

We will have the following activities:

* We will show an interactive demo in some case studies, e.g., GP to learn the ambulance dispatching policies.
* We will show a demo to explain how a policy works during the vehicle dispatch process of a vehicle routing problem.

### **Previous History**

*(If this tutorial was held in a different venue or is a modification of another tutorial, please indicate the venue and/or the changes.)*

This is the first time that we propose this tutorial.

### **Other Info**

*(Any other relevant information.)*