# Solving combinatorial optimization problems by hybrid methods combining optimization techniques with machine learning techniques

*Techin' to me*

by

Fulin Yan

*Supervisors :* Francois Clautiaux, Aurelien Froger, Boris Albar

université de BORDEAUX

Institut de Mathématiques de Bordeaux

CATIE

April 24, 2024

# Contents

# Current Section

# Combinatorial optimization

$$\min \; f(x)$$
$$\text{s.t } x \in S$$

- Any finite set and their subsets
- The set of integers $\mathbb{Z}$
- The set $\mathbb{Z}^n$ of vectors of size $n \in \mathbb{N}^*$
- Example: Knapsack problem. $\{1, \dots, n\}$ a set of $n$ items, $p_i$ profit of item $i$, $w_i$ weight of item $i$. $C$ capacity of the knapsack.
- Maximize the total profit of items in the knapsack.

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n} p_i x_i \\
\text{s.t} \quad & \sum_{i=1}^{n} w_i x_i \leq C \\
& x_i \in \{0, 1\} && \forall i \in \{1, \dots, n\}
\end{aligned}$$

# Definitions

- Feasibility: A solution $x \in S$ is feasible.
- Relaxation: Problem that some constraints of the original problem is removed
- Dual bound: Best of objective value of the relaxation
- Primal bound: objective value of a feasible solution

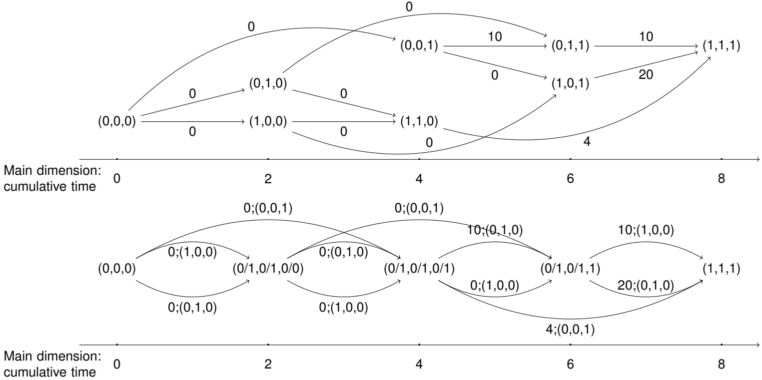
# Sequential decision processes

- Many combinatorial optimization (CO) problems can be expressed as sequential decision processes (SDP), such as knapsack problems, scheduling problems, cutting stock problems, routing problems.
- A SDP represents a succession of decisions to take based on the current state of a system.

# Sequential decision processes

- One single machine total weighted tardiness problem
  $T_i = \max(0, t_i - d_i)$

| Job | processing time $p_i$ | due time $d_i$ | weight $w_i$ |
|---|---|---|---|
| 1 | 2 | 7 | 10 |
| 2 | 2 | 4 | 6 |
| 3 | 4 | 6 | 10 |



# Pros and cons of the sequential decision processes

- Generic: SDP facilitates the adaption to new environments and new constraints.
- SDP produces mathematics formulations with interesting theoretic properties, which are exploited in a wide range of dedicated efficient algorithms.
- The formalism of SDP allows for reformulating problems as network flow problems, which is usually stronger than initial MIP formulations.
- Difficulty: the size of graphs can be pseudo-polynomial or even exponential.

# Motivation

- In the context of industrial problems, the size of the problem instances to be solved is considerable, and classical techniques of exact optimization cannot be directly applied.
- Techniques that allow for finding high-quality solutions within a limited time constraint are therefore of significant practical interest.
- Exact optimization needs also high-quality heuristic solutions.

# Motivation

- The success of machine learning (ML) in a wild range of domains
- The use of machine learning in optimization problems, for example, can help reduce the need for handcrafted heuristics in cases where it would be too difficult or costly to develop them.

# Current Section

# Research Objectives

- Developing heuristic approaches for solving SDP based on the hybridization of machine learning and combinatorial optimization

# Current Section

# Machine learning helps combinatorial optimization

- End-to-end learning [Kotary et al., 2021] [Mazyavkina et al., 2021]
- Learning to configure: [Karimi-Mamaghan et al., 2022]
- Machine learning alongside Combinatorial optimization [Bengio et al., 2021]

# End-to-end learning

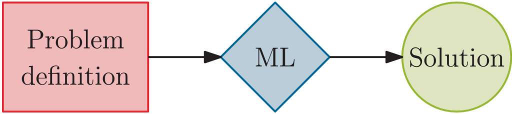



Figure 2: Machine learning acts alone to provide a solution to the problem.[Bengio et al., 2021]

- A ML model learns directly to produce solutions to CO problems.
- [Kotary et al., 2021] classify these methods into three major approaches:
    1) Learning with constraints: the features that describe problem instances are directly utilized as inputs for machine learning models to generate optimal or "good" solutions.
    2) Learning on graphs: When the problems can be represented in graph form, this approach involves learning the underlying graph structure to produce solutions.

# End to end learning

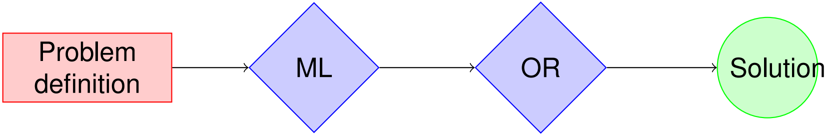



Figure 3: Predict and optimize [Dalle et al., 2022]

3) Predict and optimize: This approach involves employing a pipeline of machine learning and optimization algorithms. [Dalle et al., 2022]

# Learn to configure

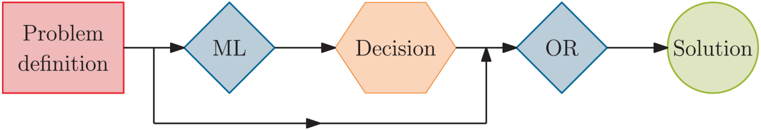



Figure 4: The machine learning model is used to augment an operation research algorithm with valuable pieces of information [Bengio et al., 2021]

- To dynamically configure the hyper-parameters of the optimization algorithm

# Machine learning alongside combinatorial optimization

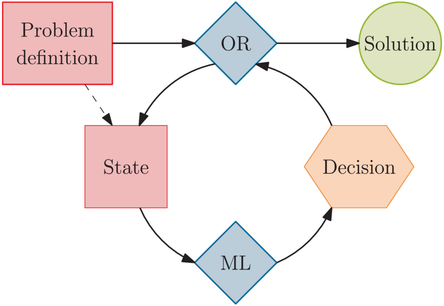



Figure 5: The combinatorial optimization algorithm repeatedly queries the same machine learning model to make decisions. The machine learning model takes as input the current state of the algorithm, which may include the problem definition [Bengio et al., 2021]

- Machine learning model is repetitively called during the optimization process to propose information helpful to guide the process.

# Current Section

# Arc-flow formulation of RCSPP

- $\mathscr{G}(\mathscr{V},\mathscr{A})$ be a directed acyclic multi-graph, where $s \in \mathscr{V}$ represents the source and $t \in \mathscr{V}$ represents the sink.
- Each arc $a \in \mathscr{A}$ has a tail $v^-(a)$ and a head $v^+(a)$.
- For each arc $a \in \mathscr{A}$, $c_a$ represents the cost of $a$.
- Let $\mathscr{R}$ be the set of resource constraints. For each resource constraint indexed by $k \in \{1\ldots,|\mathscr{R}|\}$, we have the resource capacity denoted by $r^k$, and each arc $a$ is also assigned a resource consumption $r_a^k$.
- A arc-flow formulation of RCSPP is given as follows in (1) - (4):

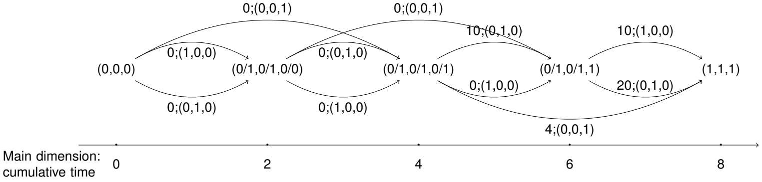
$$\min \sum_{a \in \mathscr{A}} c_a x_a \tag{1}$$

$$\text{s.t.} \sum_{a \in \mathscr{A} | v^+(a)=v} x_a - \sum_{a' \in \mathscr{A} | v^-(a')=v} x_{a'} = \begin{cases} -1, & \text{if } v=s, \\ 1, & \text{if } v=t, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall v \in \mathscr{V}, \tag{2}$$

$$\sum_{a \in \mathscr{A}} r_a^k x_a \leq (\text{or} =) r^k, \qquad \forall k \in \{1,\ldots,|\mathscr{R}|\}, \tag{3}$$

$$x_a \in \{0,1\}, \quad \forall a \in \mathscr{A} \tag{4}$$

# Resource constrained shortest path problem

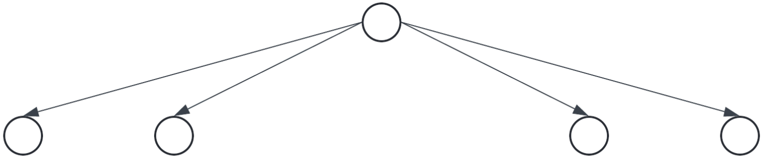



$$
\begin{aligned}
\min \quad & \sum_{a\in\mathscr{A}} c_a x_a \\
\text{s.t.} \quad & \sum_{a\in\mathscr{A}|v^{+}(a)=v} x_a - \sum_{a'\in\mathscr{A}|v^{-}(a')=v} x_{a'} = \begin{cases} -1, & \text{if } v=(0,0,0), \\ 1, & \text{if } v=(1,1,1), \\ 0, & \text{otherwise,} \end{cases} && \forall v\in\mathscr{V}, \\
& \sum_{a\in\mathscr{A}} r_a^k x_a = 1, && \forall k\in\{1,2,3\}, \\
& x_a \in \{0,1\}, \quad \forall a\in\mathscr{A}
\end{aligned}
$$

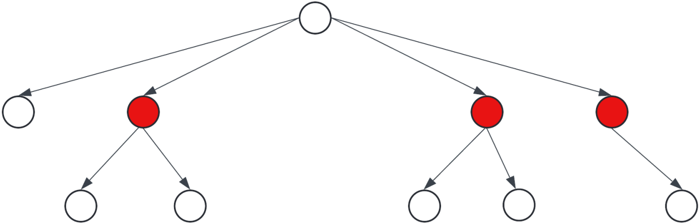
# Beam search

- A constructive heuristic, which starts with an empty solution and repeatedly extends the current partial solution until a complete solution is obtained.
- Beam search illustration

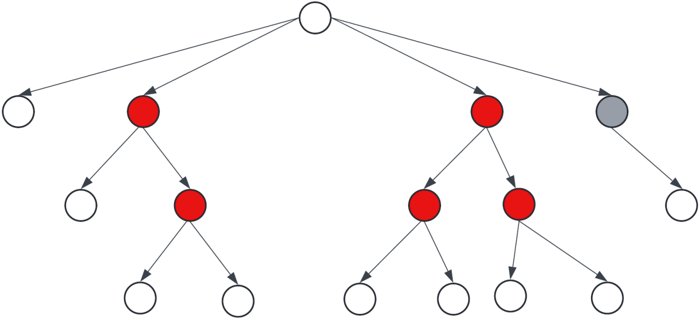
# Beam search



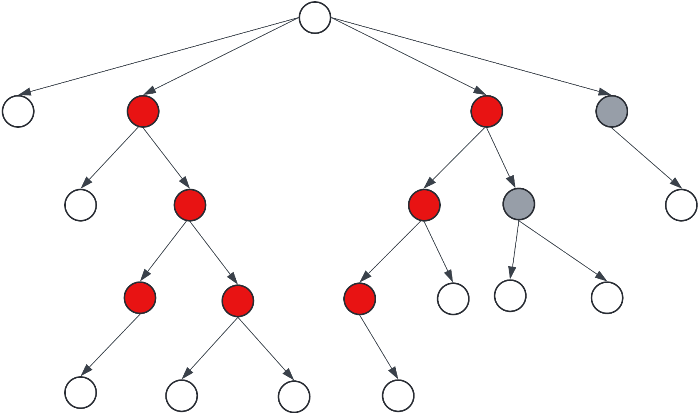
# Beam search



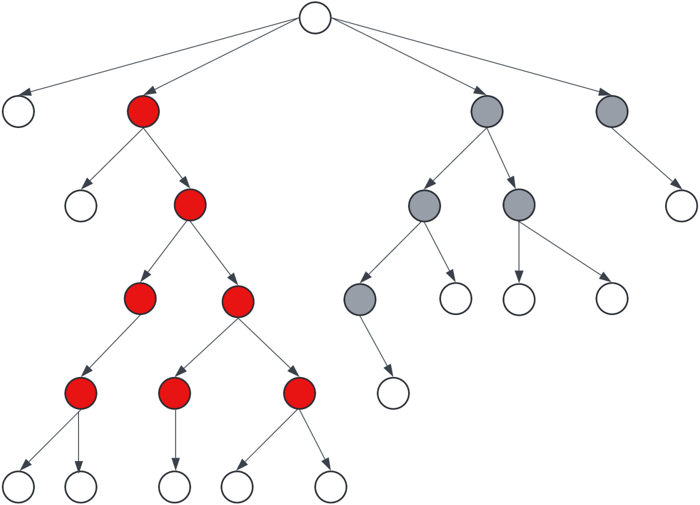
# Beam search



# Beam search



# Beam search



# Beam search

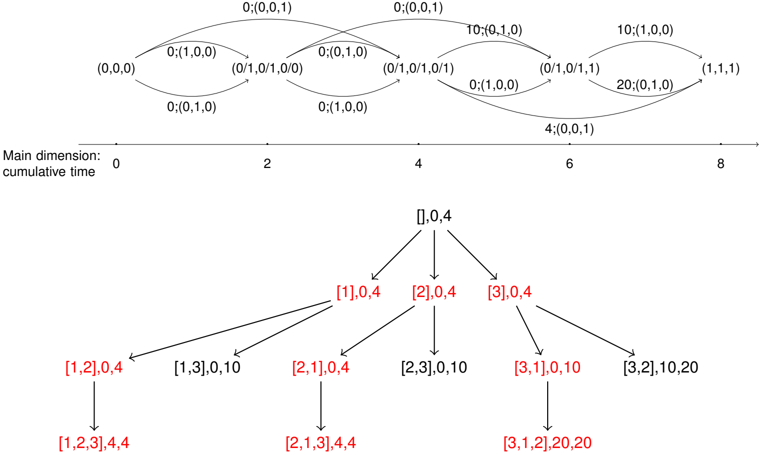
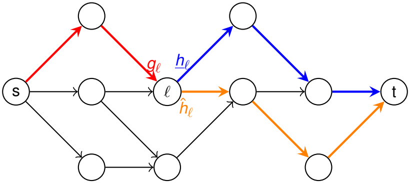



Figure 6: Beam search with beam size 3 applied to the problem $1||\sum_i w_i T_i$ above. Each node of the search tree is represented by a tuple of (partial solution, value of the partial solution, heuristic score of the partial solution)

# Beam search




- Criterion of maintaining labels: $\psi_\ell = g_\ell + h_\ell$
  - Dual bound $\psi_\ell = g_\ell + \underline{h}_\ell$: a bound given by solving the relaxation. $\underline{h}_\ell$ easy to obtain, but the quality is doubting. The completion $\underline{h}_\ell$ can be infeasible with respect to $\ell$.
  - Primal bound $\psi_\ell = g_\ell + \hat{h}_\ell$: a bound given by a incumbent solution, i.e.: the best solution so far completing a label. $\hat{h}_\ell$ is less easier to obtain, and the quality depends on the rollout heuristic. The completion $\hat{h}_\ell$ is primal, i.e.: is always feasible with respect to $\ell$.

# Beam search with machine learning alongside

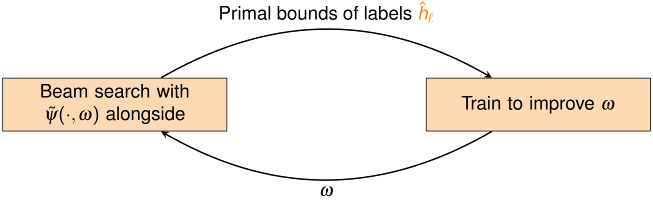
- Using a machine learning model to estimate the scoring function $\psi$.
- Let $\phi$ be a mapping from a label to a vector of $m$ features representing the label.

$$\begin{aligned}\phi :& \mathscr{L} \to \mathbb{R}^m \\ &\ell \to \phi(\ell)\end{aligned}$$

- Let $\tilde{\psi}(\cdot;\omega)$ be a function parameterized by $\omega$ such that $\psi(\ell;\omega) = \tilde{\psi}(\phi(\ell);\omega)$

$$\begin{aligned}\tilde{\psi}(\cdot;\omega) :& \mathscr{R}^m \to \mathbb{R} \\ &\phi(\ell) \to \psi_\ell(\omega) = \tilde{\psi}(\phi(\ell);\omega)\end{aligned}$$

- The objective is to train $\tilde{\psi}(\cdot;\omega)$ such that $\psi_\ell(\omega)$ estimates the primal bound of solutions completing $\ell$. [Huber and Raidl, 2022]



# Preliminary results without learning alongside

- 125 instances of $1||\sum w_i T_i$ with 100 jobs.
- The preliminary results show the solution quality of beam search (BS) using the best dual bound $v_\ell$ as the heuristic criterion without ML model alongside compared to the baseline.
- Baseline: early due date (EDD) + shortest processing time (SPT) + local search (LS)
- gap = (val_heur - val_baseline)/val_heur $\times$ 100%

| Heuristic | beam size | meanGap(%) | stdGap(%) |
|---:|---:|---:|---:|
| EDD+SPT+LS | - | - | - |
| EDD | - | 55.11 | 18.52 |
| SPT | - | 62.42 | 29.40 |
| BS | 10 | 2.23 | 3.41 |
| | 20 | 1.57 | 2.67 |
| | 100 | 1.00 | 1.90 |
| | 200 | 0.86 | 1.76 |
| | 500 | **0.66** | **1.39** |

# How to train the ML model

1. Pre-training phase:
   1.1 Run the beam search using **the best dual bound** $\nu_\ell$ as the heuristic criterion to generate labels and to obtain the primal completion $\hat{h}_\ell$.
   1.2 Train $\tilde{\psi}(\cdot;\omega)$ using targets $\psi_\ell = g_\ell + \hat{h}_\ell$.
2. Training Phase:
   2.1 Run the beam search using **the ML output** $\tilde{\psi}(\cdot;\omega)$ as the the heuristic criterion to generate labels and to obtain the primal completion $\hat{h}_\ell$
   2.2 Train $\tilde{\psi}(\cdot;\omega)$ using targets $\psi_\ell = g_\ell + \hat{h}_\ell$.
   2.3 If the stop criterion is not satisfied, go to 2.1

# Current Section

# Important References

[Bengio et al., 2021] Bengio, Y., Lodi, A., and Prouvost, A. (2021).
Machine learning for combinatorial optimization: A methodological tour d'horizon.
*European Journal of Operational Research*, 290(2):405–421.

[Dalle et al., 2022] Dalle, G., Baty, L., Bouvier, L., and Parmentier, A. (2022).
Learning with Combinatorial Optimization Layers: A Probabilistic Approach.

[Huber and Raidl, 2022] Huber, M. and Raidl, G. R. (2022).
Learning Beam Search: Utilizing Machine Learning to Guide Beam Search for Solving Combinatorial Optimization Problems.
In Nicosia, G., Ojha, V., La Malfa, E., La Malfa, G., Jansen, G., Pardalos, P. M., Giuffrida, G., and Umeton, R., editors, *Machine Learning, Optimization, and Data Science*, Lecture Notes in

# Important References

Computer Science, pages 283–298. Springer International Publishing.

[Karimi-Mamaghan et al., 2022] Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., and Talbi, E.-G. (2022).
Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art.
*European Journal of Operational Research*, 296(2):393–422.

[Kotary et al., 2021] Kotary, J., Fioretto, F., Van Hentenryck, P., and Wilder, B. (2021).
End-to-End Constrained Optimization Learning: A Survey.

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

Thank You