Learning to Generate Sub-Problems in Mixed Integer Programming

Luca Mossina, PhD student[1]

[1]Dept. Complex Systems Engineering, ISAE-Supaero School of Engineering.
University of Toulouse, France.

Introduction

- Resolution of **recurrent combinatorial optimization** problems, coupling **machine learning** techniques with **branch & bound** algorithms, operating under a **limited time budget**.
- Assuming problems are the realization of a generative process, historical data are collected and used to train a **classification model**.
- At first, when solving a new instance, this model will select a **subset** of decision **variables** to be set heuristically to some **reference values**, becoming **fixed parameters**.
- The **remaining variables** are left free and form a **smaller sub-problem** whose solution, while being an approximation of the optimum, can be obtained sensibly faster.
- Subsequently, if some of the time allocated is available, an iterative process of **blocking/unblocking variables** takes place, allowing to explore other areas of the solution space.
- This approach is of particular interest for problems where perturbations on the instance parameters can occur unexpectedly, requiring a rapid re-optimization of a complex model.

Overview & Example

Example: problem P is one of energy production planning. All days are similar, the network is mostly unchanged, only the energy demands vary: **recurrent** combinatorial optimization.
$\longrightarrow$ an *anomaly* occurs, some parameters can change: new problem P', re-optimization is necessary.
Response time is limited: resolution **time budget**.

$\longrightarrow$ Variations on a theme: generative process
$\longrightarrow$ Past resolutions, data available
$\longrightarrow$ Learn from the past, solve future instances faster

Methods developed:

(1) ***NaiBRec*** : **block** irrelevant variables, only one SP generation.
Given $P' \xrightarrow{\text{generate}}$ SP: smaller sub-problem, manageable approximation.

(2) ***SuSPen*** : **block/unblock** variables, iterative SP generation.
Given $P \xrightarrow{\text{generate}}$ SP **repeatedly**, until resolution time is available.

Blocked variables are assigned a value **heuristically**. E.g., the optimal solution x^*_{ref} to a reference problem P_{ref}, such as the model under *nominal parameters*

Mixed Integer Programming

- A mixed integer programming (MIP) Problem P is:

$$z^* = \min \{c^t x \quad | \, x \in \chi\}$$

where $\chi = \{x \in \mathbb{R}^p \times \mathbb{Z}^q, \text{ s.t. } Ax \leqslant b\}$.

- We define a sub-problem SP of P as:

$$z_{SP} = \min \{c^t x_{SP} + \underbrace{\boxed{K}}_{\text{const}} \mid x_{SP} \in \chi\}$$

$\longrightarrow$ **block** a (set of) variables to a **reference value**: $x_B = \{x^j \mid x := x_{Ref}\}$.
$\longrightarrow$ $K = c^t_B x_B$: **constant** in the new problem SP.
$\longrightarrow$ $x_{SP} = x \setminus x_B$ are the remaining **free** variables.

MIP is a widely adopted formulation for combinatorial optimization problems, solved via Branch & Bound algorithms. Highly performant solvers exist (CPLEX, GUROBI), for reasonably sized instances.

$\Longrightarrow$ intrinsic exponential complexity: won't go away.

(1) NaiBRec: Naive Bayes for Recurrent Problems

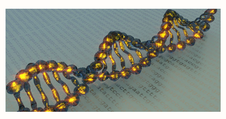
How to block variables? Frame as a multi-label classification problem, find variables not affected by random events.

Multi-label Classification (MLC)

- **Image annotation**:

$\mapsto \{\text{beautiful}, \text{mountains}, \text{Dolomites}, \text{go}, \text{holiday}, \text{summer}, \text{winter}\}$

- **Genetic data**: classification of gene functions.



$\mapsto \{\text{cell growth}, \text{cell multiplication}, \text{structural function}\}$

$$\{\text{labels}\} \equiv \{\text{decision variables}\}$$

$\Longrightarrow$ *How many* **labels** to include? $\equiv$ *How many* **variables** to block?
$\Longrightarrow$ *Which* **labels** to choose? $\equiv$ *Which* **variables** to block?

Note: when variables are too many, use **clusters** and **hierarchies** of variables as **proxy** labels.

NaiBX MLC Algorithm – Cascade of Predictors

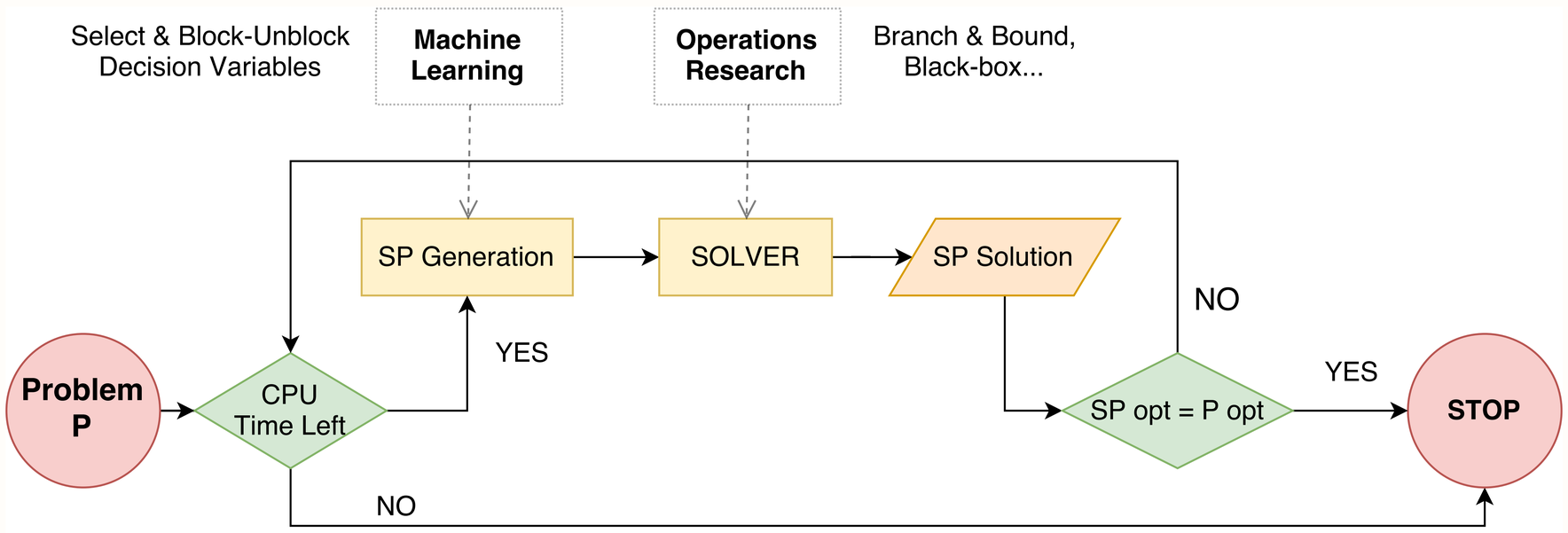
Step (1) Size estimation, m: number of labels to be predicted
Step (2) Sequential label prediction, given the size of the target vector
$m \rightarrow y_{(1)} \rightarrow y_{(2)} \rightarrow \ldots \rightarrow y_{(m-1)} \rightarrow y_{(m)}$.
$\rightarrow$ Each step is a *naive Bayes* classifier.

NaiBRec Meta-Algorithm

Step (1) Collect past data, train classification model.
Step (2) Extract features from current P', e.g. differences $P' \leftrightarrow P_{ref} \Rightarrow \Delta(MIP - parameters)$.
Step (3) $h_{NaiBX} : \Delta(\text{parameters}) \longrightarrow \{x_B, x_{SP}\}$, predict variables to be blocked.
Step (4) Generate SP
Step (5) Run optimization

(2) SuSPen, Supervised Sub-Problem Generation

Extends NaiBRec, introducing the concept **blocking/unblocking** decision variables. It explores the solution space while handling small problems. It respects the time constraint imposed.



Perspectives

ROTEx: Reinforced Optimization Tree Exploration.

- Considering the block/unblock process as a Markov Decision Process (MDP).
- **Reinforcement Learning** framework.

Acknowledgments

This research benefited from the support of the *"FMJH Program Gaspard Monge in optimization and operations research"* and from the support to this program from *EDF*.