Adaptive Cut Selection report

Daniele Ferrario

January 2025

1 Introduction

Here we sum up the experience of running the experiments presented by Mark Turner on Adaptive Cut Selection in Mixed-Integer Linear Programming.

2 Main requirements

1. SCIP: *SCIP is currently one of the fastest non-commercial solvers for mixed integer programming (MIP) and mixed integer nonlinear programming (MINLP). It is also a framework for constraint integer programming and branch-cut-and-price. It allows for total control of the solution process and the access of detailed information down to the guts of the solver."*
2. SLURM: installed on the HPC systems Iris and Aion, *is a free and open-source job scheduler for Linux and Unix-like kernels, used by many of the world's supercomputers and computer clusters.*
3. PySCIPOpt: A wrapper for SCIP providing a in interface for python

3 Datasets

- **MIPLIB2017**
- A NN-verification dataset, to evaluate the robustness of the model we will train.

4 Standard data generation

To conduct our analysis, we want to generate standard data from the original instances, meaning:

1. *.mps* the problem after a **presolve** stage (do it just once for all following experiments). They are called "Transformed problems"
2. *.sol* file for every instance
3. *.yml* file containing solving quality measures with default cut selector parameter values
4. *.log* file showing all output from SCIP solver

If the number of instances is relatively high, the experiment could crash without any warning. This has been solved by using the *bigmem* node on the cluster.

5 Grid search

The program tries all the possible combinations out of the 286 variable choices for the scoring rule, with granularity of 0.1 and sum 1.0, so that:

$$\sum_{i=1}^{4} \lambda_i = 1,\ \lambda_i = \frac{\beta_i}{10}, \beta_i \in N, \quad \forall i \in \{1, 2, 3, 4\}$$

This step can also be run multiple times with respect to different seeds of choice. (Takes a long time)

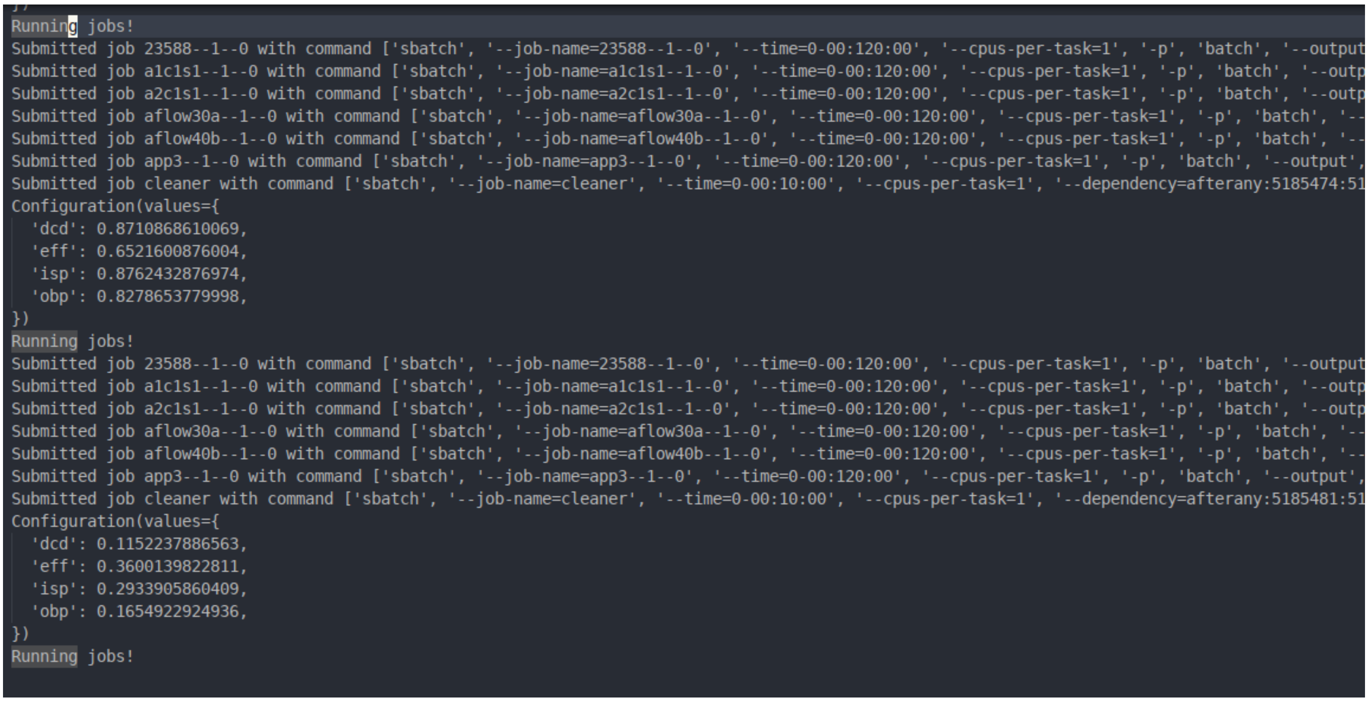
```yaml
a2c1s1:
  improvement: 0.35316249260417526
  parameters:
  - - 0.4
    - 0.3
    - 0.3
    - 0.0
aflow30a:
  improvement: 0.04108450936279827
  parameters:
  - - 0.0
    - 0.8
    - 0.1
    - 0.1
```

Listing 1: Example of grid search on these two instances

Grid search will act as the base result for the following experiments.

6 SMAC

Before diving into the neural network approach, the author wants to evaluate another standard ML approach which is **SMAC**. The smac package in Python is a Sequential Model-based Algorithm Configuration framework. It is primarily used for hyperparameter optimization and algorithm configuration using Bayesian optimization. Unlike the other approaches, which return instance- dependent cut selector parameters, SMAC will return a single set of parameter values that works for every instance over the entire instance set.



```
Running jobs!
Submitted job 23588--1--0 with command ['sbatch', '--job-name=23588--1--0', '--time=0-00:120:00', '--cpus-per-task=1', '-p', 'batch', '--output
Submitted job a1c1s1--1--0 with command ['sbatch', '--job-name=a1c1s1--1--0', '--time=0-00:120:00', '--cpus-per-task=1', '-p', 'batch', '--outp
Submitted job a2c1s1--1--0 with command ['sbatch', '--job-name=a2c1s1--1--0', '--time=0-00:120:00', '--cpus-per-task=1', '-p', 'batch', '--outp
Submitted job aflow30a--1--0 with command ['sbatch', '--job-name=aflow30a--1--0', '--time=0-00:120:00', '--cpus-per-task=1', '-p', 'batch', '--
Submitted job aflow40b--1--0 with command ['sbatch', '--job-name=aflow40b--1--0', '--time=0-00:120:00', '--cpus-per-task=1', '-p', 'batch', '--
Submitted job app3--1--0 with command ['sbatch', '--job-name=app3--1--0', '--time=0-00:120:00', '--cpus-per-task=1', '-p', 'batch', '--output',
Submitted job cleaner with command ['sbatch', '--job-name=cleaner', '--time=0-00:10:00', '--cpus-per-task=1', '--dependency=afterany:5185474:51
Configuration(values={
  'dcd': 0.8710868610069,
  'eff': 0.6521600876004,
  'isp': 0.8762432876974,
  'obp': 0.8278653779998,
})
Running jobs!
Submitted job 23588--1--0 with command ['sbatch', '--job-name=23588--1--0', '--time=0-00:120:00', '--cpus-per-task=1', '-p', 'batch', '--output
Submitted job a1c1s1--1--0 with command ['sbatch', '--job-name=a1c1s1--1--0', '--time=0-00:120:00', '--cpus-per-task=1', '-p', 'batch', '--outp
Submitted job a2c1s1--1--0 with command ['sbatch', '--job-name=a2c1s1--1--0', '--time=0-00:120:00', '--cpus-per-task=1', '-p', 'batch', '--outp
Submitted job aflow30a--1--0 with command ['sbatch', '--job-name=aflow30a--1--0', '--time=0-00:120:00', '--cpus-per-task=1', '-p', 'batch', '--
Submitted job aflow40b--1--0 with command ['sbatch', '--job-name=aflow40b--1--0', '--time=0-00:120:00', '--cpus-per-task=1', '-p', 'batch', '--
Submitted job app3--1--0 with command ['sbatch', '--job-name=app3--1--0', '--time=0-00:120:00', '--cpus-per-task=1', '-p', 'batch', '--output',
Submitted job cleaner with command ['sbatch', '--job-name=cleaner', '--time=0-00:10:00', '--cpus-per-task=1', '--dependency=afterany:5185481:51
Configuration(values={
  'dcd': 0.1152237886563,
  'eff': 0.3600139822811,
  'isp': 0.2933905860409,
  'obp': 0.1654922924936,
})
Running jobs!
```

Figure 1: SMAC iterations

7 Generate GCNN input and train

Calling this function will create *coefficients.npy*, *col_features.npy*, *edge_indices.npy*, *row_features.npy* for each instance of the transformed problems, where these files are used to construct the input into our graph neural network. File extension refers to Numpy objects.

7.1 Training and testing

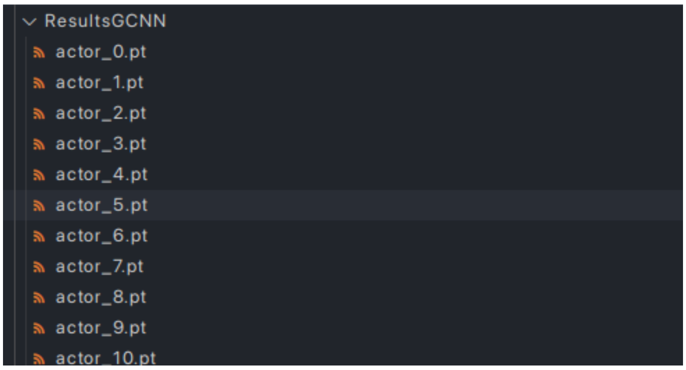


Figure 2: Progressively saved GCNN files

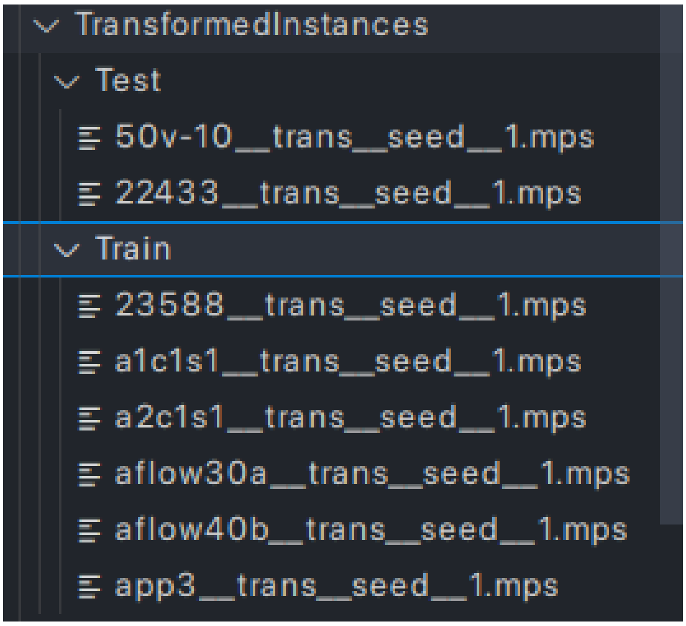


Figure 3: Test and train instances