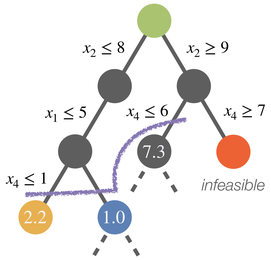
Learning to Schedule Heuristics in Branch and Bound

Antonia Chmiela chmiela@zib.de
Elias B. Khalil khalil@mie.utoronto.ca
Ambros Gleixner gleixner@zib.de
Andrea Lodi andrea.lodi@polymtl.ca
Sebastian Pokutta pokutta@zib.de

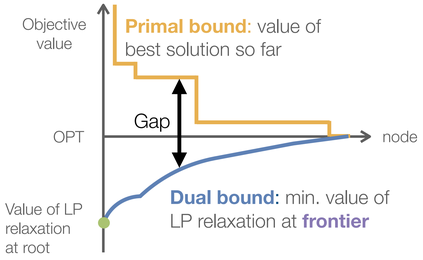
NEURAL INFORMATION PROCESSING SYSTEMS

Solving MIPs with Branch and Bound

- Mixed Integer Programs (MIPs) are optimization problems with additional constraints that some of the variables need to be integer
- commonly solved with the Branch and Bound algorithm: an optimal solution is found by successively dividing the problem into smaller linear subproblems



BRANCH AND BOUND TREE

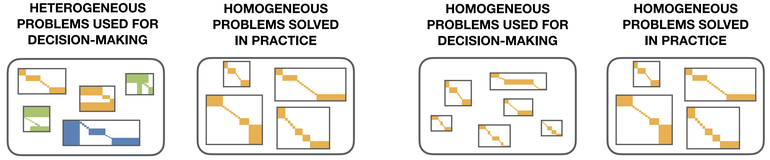


SOLVING PROCESS

- primal heuristics are crucial for finding good primal solutions → helps to close the gap faster and enables more effective decision-making in practise
- typically, solvers use a lot of different heuristics with a multitude of parameters

The Problem with Managing Heuristics

- most heuristics can be really costly → important to be strategic about setting their working limits
- solvers make such decisions following hard-coded rules derived from broad benchmark test sets, but heuristic's performance is highly problem-dependent
- *PROBLEM:* Static settings derived from heterogeneous benchmark sets do not yield best performance for specific problem class

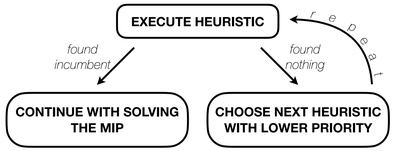


WHAT SOLVERS DO **WHAT USERS WANT**

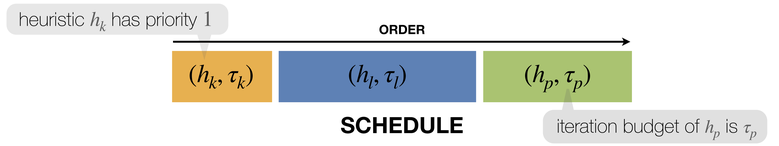
- *SOLUTION:* Use a data-driven framework to learn how to control primal heuristics perfectly for the problem class of interest

What to Control?

- **heuristic loop** iterates over heuristics until good enough incumbent is found
- each heuristic has limits that decide **for how long** it will be executed
- *IDEA:* Control order and duration to find better solutions faster → build **schedule of heuristics**

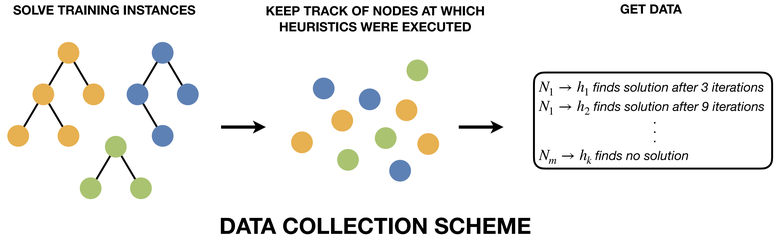


HEURISTIC LOOP



SCHEDULE

Learning from Data



DATA COLLECTION SCHEME

- *IDEA:* learn from data by constructing schedule that finds solutions for large fraction of nodes while minimizing number of iterations spent by schedule

The Heuristic Scheduling Problem

number of iterations schedule S needs to solve node N

nodes solved by schedule S

$$\min_{S} \sum_{N \in N(X)} T(S,N) \quad \text{s.t.} \quad |N(S)| \geq \alpha |N(X)|$$

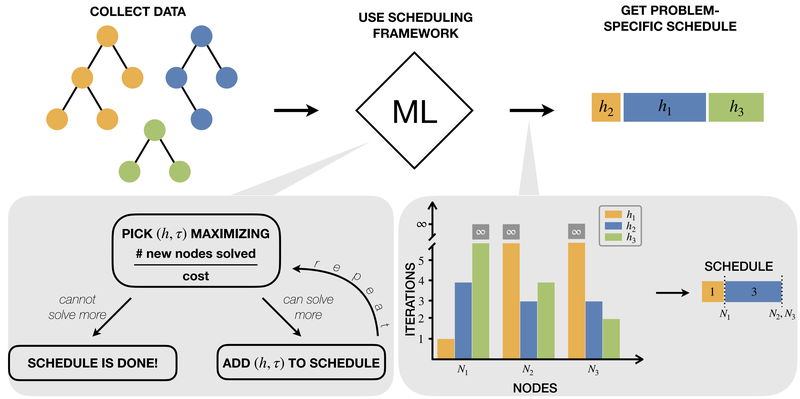
nodes that appear when solving instances of a training set X

minimum fraction of nodes at which S needs to find a solution

- problem is NP-hard since it generalizes the *Pipelined Set Cover Problem*

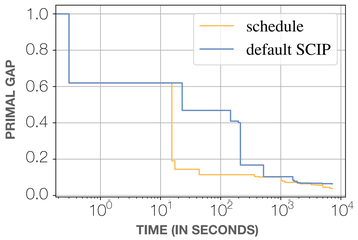
Obtaining a Schedule Efficiently

- *IDEA:* Build **greedy schedule** by successively adding actions (h, τ) that maximize the benefit-cost ratio, i.e., number of nodes solved to total cost.
- complexity: $O(|\text{Dataset}|)$

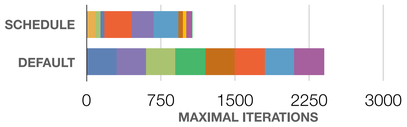


Computational Results

- we **consistently** learn schedules with **better primal performance** than default settings of state-of-the-art academic MIP solver SCIP
- tested on challenging problem classes: *Generalized Independent Set Problem (GISP)* and *Fixed-Charge Multi-Commodity Network Flow Problem (FCMNF)*



	PRIMAL INTEGRAL IMPROVEMENT	INSTANCE WITH BETTER PRIMAL INTEGRAL	INSTANCES WITH BETTER PRIMAL BOUND	INCREASE IN INCUMBENTS FOUND
GISP	49 %	92 %	57 %	22 %
FCMNF	14 %	73 %	78 %	129 %



- schedule uses different order and are **much shorter** → intelligent heuristic scheduling **increases** probability of finding solutions
- since the **learning phase is efficient, effective and scalable**, training is much faster than other ML approaches