Efficient Primal Heuristics for Mixed-Integer Linear Programs

Linxin Yang, Sha Lai, Akang Wang, Xiaodong Luo

Shenzhen Research Institute of Big Data, China

Xiang Zhou, Haohan Huang, Shengcheng Shao, Yuanming Zhu, Dong Zhang

Algorithm & Technology Development, Service & Software R&D Management, GTS, Huawei

NeurIPS 2021 Annual Conference

December 9, 2021

Mixed-Integer Linear Program

$$\begin{aligned} \min_{x,y} \quad & c^\top x + d^\top y \\ \text{s.t.} \quad & Ax + By \leq h \\ & x \in \mathbb{R}^n, y \in \mathbb{Z}^m \end{aligned}$$

Parameters (c, d, A, B, h) follow some distribution.

Question

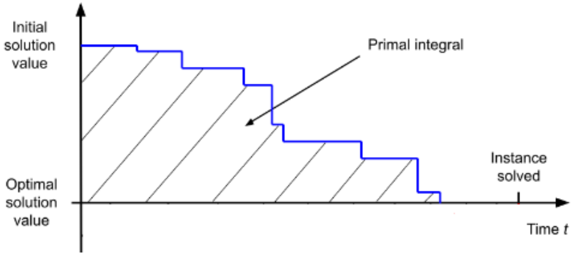
Can we utilize machine learning (ML) to enhance/speed up the optimization step?

✓ In this work, we do not utilize ML to design primal heuristics but rely on it for tuning parameters of our proposed heuristics.

ML4CO Competition

Three tasks:

- primal task: primal heuristics at the root node
- dual task: variable selection for branching
- configuration task: parameter tuning before solving



Final Results: Primal Task

Three datasets:

- item placement
- load balancing
- anonymous

Primal task

rank	team	item_placement cum. reward	item_placement rank	load_balancing cum. reward	load_balancing rank	anonymous cum. reward	anonymous rank	score
1	CUHKSZ_ATD	-3355.56	1	-213467.31	1	-45747908.15	1	1
2	UNIST-LIM-Lab	-5902.27	5	-214696.55	2	-48690877.94	2	20
3	MDO	-4798.50	3	-216053.14	3	-52610680.40	3	27
4	EI-OROAS	-4099.32	2	-217743.57	4	-101312897.27	4	32
5	Mr_Tree	-5988.72	7	-218451.95	5	-112201967.54	5	175

Item Placement

- $i \in I := \{0, 1, ..., 104\}$, the set of items
- $j \in J := \{0, 1, ..., 9\}$, the set of containers
- $k \in K := \{0, 1, 2\}$, the set of dimensions

Item Placement

Consider a multi-dimensional multi-knapsack model:

$$
\begin{aligned}
\min_{x,y,z} \quad & \sum_{j\in J}\sum_{k\in K} \alpha_k y_{jk} + \sum_{k\in K} \beta_k z_k \\
\text{s.t.} \quad & \sum_{j\in J} x_{ij} = 1 && \forall i \in I \\
& \sum_{i\in I} a_{ik} x_{ij} \le b_k && \forall j \in J, \forall k \in K \\
& \sum_{i\in I} d_{ik} x_{ij} + y_{jk} \ge 1 && \forall j \in J, \forall k \in K \\
& y_{jk} \le z_k && \forall j \in J, \forall k \in K \\
& x_{ij} \in \{0,1\} && \forall i \in I, \forall j \in J \\
& y_{jk} \ge 0 && \forall j \in J, \forall k \in K
\end{aligned}
$$

Item Placement: Statistics

深圳市大数据研究院 Shenzhen Research Institute of Big Data HUAWEI

Across all instances, we collect the following statistics.

parameters	$i \in \{0, 1, ..., 99\}$			$i \in \{100, 101, ..., 104\}$		
	range	mean	std	range	mean	std
a_{ik}	$(0, 0.1]$	0.05	0.0008	$[0.9, 1.0]$	0.95	0.0008
d_{ik}	$(0, 0.11]$	0.051	0.0008	$[0.83, 1.15]$	0.97	0.0015
a_{ik}/d_{ik}	$[0.86, 1.10]$	0.97	0.0008	$[0.86, 1.10]$	0.97	0.0008

- $b_k \in [1.72, 2.20]$, mean $= 1.95$, std $= 0.003$.
- $\alpha_k = 1, \beta_k = 300$

Observation

Since a_{ik}, d_{ik} values of items $100 - 104$ are significantly larger than those of other items, we can apply symmetry-breaking constraints $x_{100,0} = x_{101,1} = x_{102,2} = x_{103,3} = x_{104,4} = 1$.

Item Placement: Algorithms

深圳市大数据研究院 Shenzhen Research Institute of Big Data HUAWEI

- Meta-heuristics:
  - Greedy heuristics for knapsack problems
  - Large neighborhood search (LNS)
- Math-heuristics:
  - Two-step approach: (i) assign items to the first five containers by solving an assignment model defined as follows:
    (a) combine small items to generate candidates that might be placed in the first five containers;
    (b) assume all items that are placed at the remaining containers have "continuous" sizes (rather than "discrete");
    (c) add an SOS1 constraint for each small item.

    (ii) assign the remaining items to the last five containers by solving a sub-MIP.
  - Fix-and-optimize: properly choose two out of the last five containers, and then solve a sub-MIP to reassign items within those two containers optimally.

Load Balancing

- $i \in I := \{0, 1, ..., 59\}$, the set of tasks
- $j \in J := \{0, 1, ..., 999\}$, the set of machines
- $J^i \subseteq J$, a subset of machines that are accessible by task i

深圳市大数据研究院 Shenzhen Research Institute of Big Data HUAWEI

Consider a load balancing formulation:

$$
\begin{aligned}
\min_{x,y} \quad & \sum_{j\in J} y_j \\
\text{s.t.} \quad & x_{ij} \le a_i y_j & \forall i \in I, \forall j \in J^i \\
& \sum_{i\in I: j\in J^i} x_{ij} \le b_j & \forall j \in J \\
& \sum_{j\in J^i\setminus\{j'\}} x_{ij} \ge a_i & \forall i \in I, \forall j' \in J^i \\
& y_j \in \{0,1\} & \forall j \in J \\
& 0 \le x_{ij} \le b_j & \forall i \in I, \forall j \in J^i
\end{aligned}
\tag{\#}
$$

Observation

Rounding up a fractional solution would always produce a feasible solution.

Load Balancing: Strengthening

Based on the statistics across all instances, we notice $b_j < a_i$. As a result, model (#) is equivalent to the following:

$$
\begin{aligned}
\min_{x,y} \quad & \sum_{j \in J} y_j \\
\text{s.t.} \quad & 0 \le x_{ij} && \forall i \in I, \forall j \in J^i \\
& \sum_{i \in I : j \in J^i} x_{ij} \le b_j y_j && \forall j \in J \\
& \sum_{j \in J^i \setminus \{j'\}} x_{ij} \ge a_i && \forall i \in I, \forall j' \in J^i \\
& y_j \in \{0, 1\} && \forall j \in J
\end{aligned}
\qquad (\star)
$$

Note that $(\star)$ is much smaller and has a tighter LP relaxation than (#).

Load Balancing: Algorithms

- Rounding heuristics: to balance feasibility and optimality, we combine binary search with quantile selection to choose good rounding thresholds for y.
  - the root LP solution
  - the optimal LP solution to $(\star)$
- RENS [Ber14]: we define and solve a sub-MIP based on $(\star)$ and its LP solution

Key Challenges:

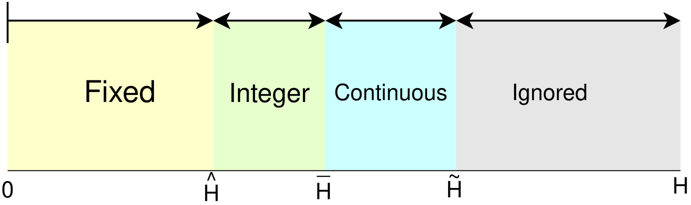
- model formulations are not exactly the same (at least from our observation)
- model size changes

Observation

One can define a planning horizon (denoted by H) and associate with each discrete variable a time period (denote by h).

- Feasibility pump [FGL05, BFL07]: apply rounding + projection + weak/strong perturbation to generate the first incumbent
- RENS: define a sub-MIP by fixing those discrete variables with their h values less than $0.9H$ at the incumbent and then solve it with a time limit.
- Rolling-horizon:
  - (i) ignore constraints involving discrete variables with their h values greater than $\tilde{H}$;
  - (ii) relax the integrality constraints on variables with their h values greater than $\bar{H}$;
  - (iii) fix those discrete variables with their h values less than $\hat{H}$ (note $\hat{H} < \bar{H} < \tilde{H}$) at the optimal solution from a previous run;
  - (iv) solve the sub-MIP;
  - (v) increase $\hat{H}, \bar{H}$ and $\tilde{H}$ adaptively and then iterate procedure (i) - (iv).

Anonymous: Rolling-horizon



Summary

	Modeling	Construction	Improvement
item	sym. break.	greedy assign. + sub-MIP	adaptive LNS fix-and-optimize
load	reformulation	data-driven rounding	RENS
anon.		feasibility pump	sequence-informed RENS rolling-horizon w. var. windows

References I

[Ber14] Timo Berthold, *Rens*, Mathematical Programming Computation **6** (2014), no. 1, 33–54.

[BFL07] Livio Bertacco, Matteo Fischetti, and Andrea Lodi, *A feasibility pump heuristic for general mixed-integer problems*, Discrete Optimization **4** (2007), no. 1, 63–76.

[FGL05] Matteo Fischetti, Fred Glover, and Andrea Lodi, *The feasibility pump*, Mathematical Programming **104** (2005), no. 1, 91–104.