FROM A BRANCH-AND-BOUND TO A STATE-OF-THE-ART HEURISTIC

An example on the permutation flowshop problem

Pablo Andres Focke[2], Vincent Jost[2], **Luc Libralesso**[1], Aurélien Secardin[2]

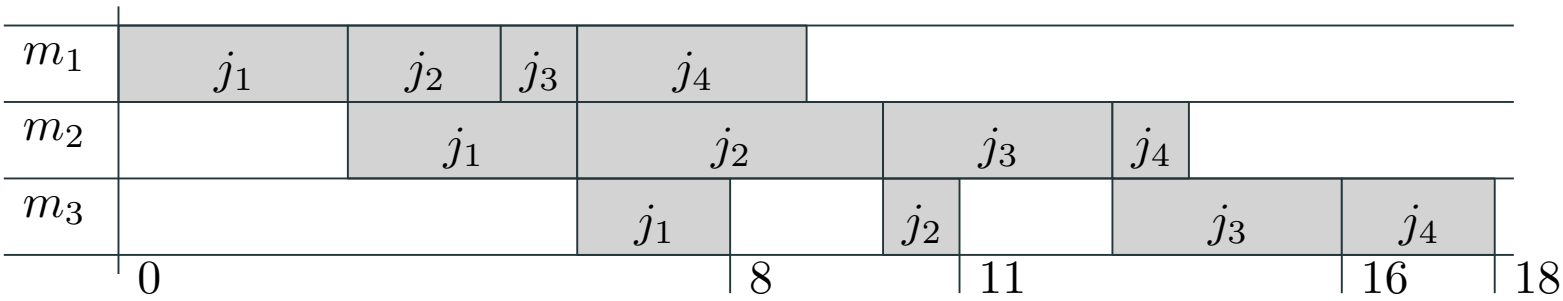
May, 6, 2021

1. LIMOS, Clermont-Ferrand, France
2. G-SCOP, Grenoble, France

email: **luc.libralesso@uca.fr**

The permutation flowshop

Famous and fundamental problem



We study 2 objectives:

The makespan: completion time of the last job on the last machine

obj: 18

total completion time (flowtime): sum of completion times

obj: 8+11+16+18 = 53

The benchmarks

Makespan:

Taillard: sizes between *20 x 5* and *500 x 20*

VFR small: sizes between *10 x 5* and *60 x 20*

VFR large: sizes between *100 x 20* and *800 x 60*

Total completion time:

Taillard: sizes between *20 x 5* and *500 x 20*

Resolution methods

exact methods: mostly **branch-and-bounds**

Gmys et al. [2020], Tomazella and Nagano [2020]

meta-heuristics: mostly **iterated-greedy algorithms**

Fernandez-Viagas and Framinan [2019]

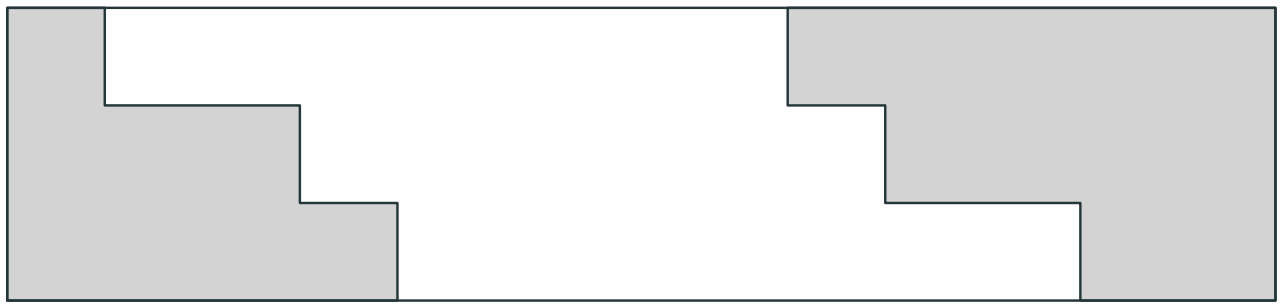
Pagnozzi and Stützle [2019]

Kizilay et al. [2019]

Focus on branch-and-bounds: bi-directional branching

Rationale: add jobs at the beginning and at the end

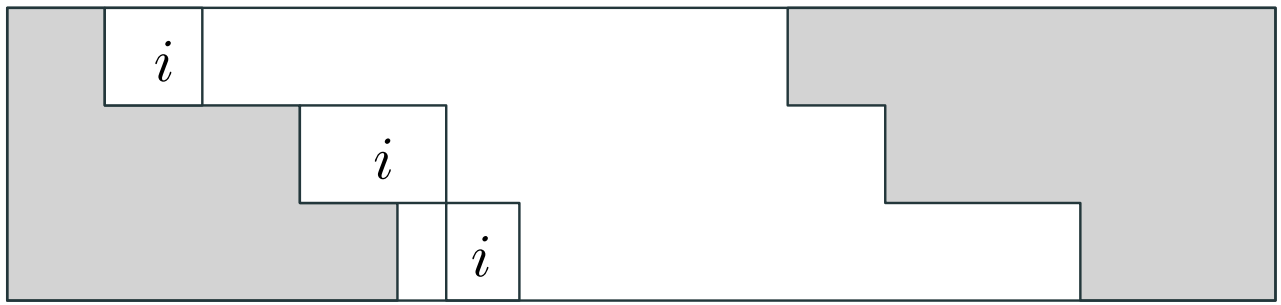
1. choose the insertion direction (forward or backward)
2. branch on jobs



Focus on branch-and-bounds: bi-directional branching

Rationale: add jobs at the beginning and at the end

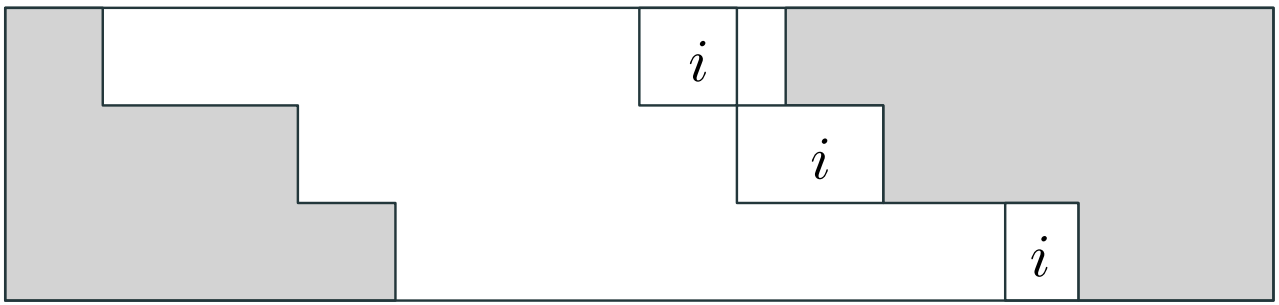
1. choose the insertion direction (forward or backward)
2. branch on jobs



Focus on branch-and-bounds: bi-directional branching

Rationale: add jobs at the beginning and at the end

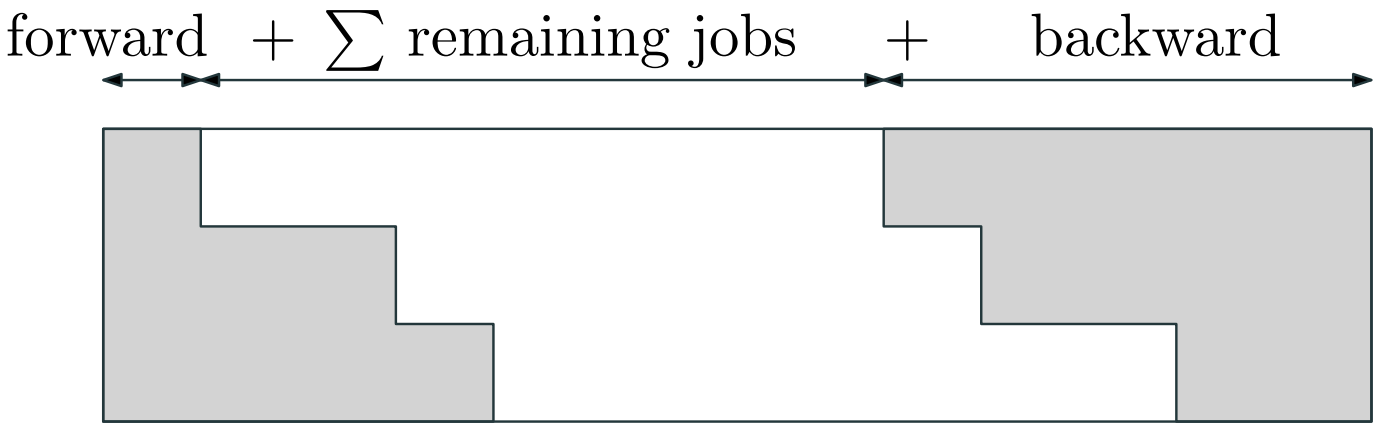
1. choose the insertion direction (forward or backward)
2. branch on jobs



Focus on branch-and-bounds: bi-directional branching

Rationale: add jobs at the beginning and at the end

1. choose the insertion direction (forward or backward)
2. branch on jobs



How to transform this branch-and-bound into an heuristic?

Key idea: a branch-and-bound explores a search-space

can we focus on “a-priori” good solutions first? (spoiler: yes)

How to transform this branch-and-bound into an heuristic?

I. The search strategy
II. The guidance strategy

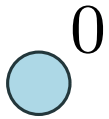
The "classical" search strategy: Depth First Search

- small memory requirements
- simple

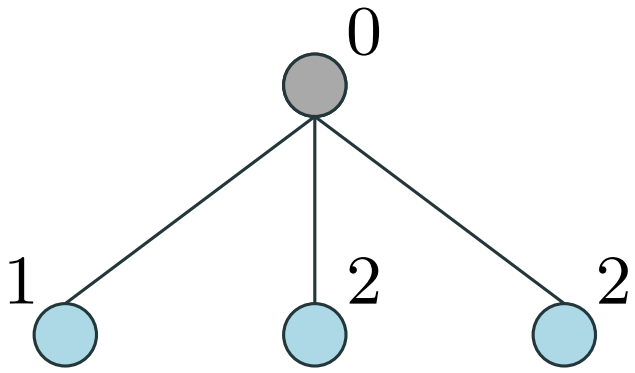
BUT

- gets usually stuck in "early bad decisions"

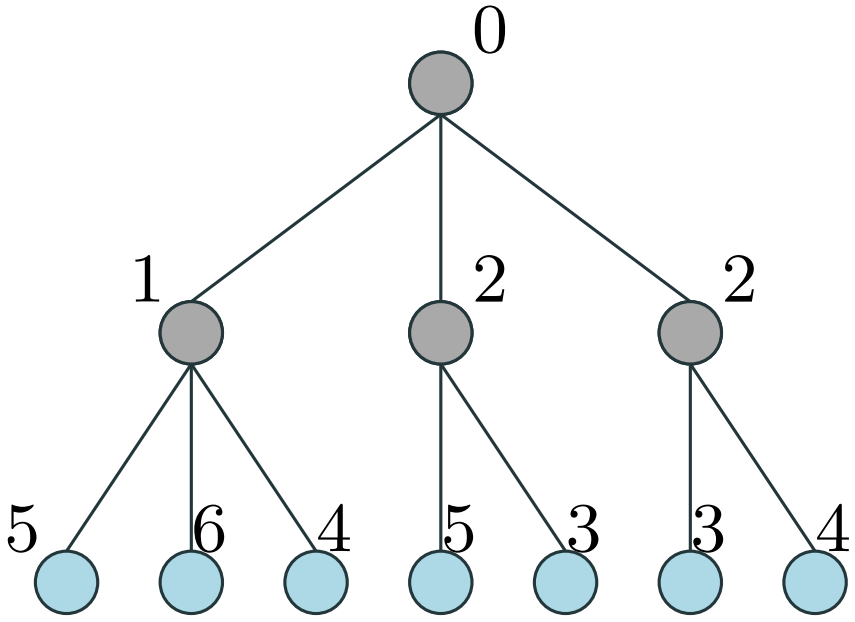
Another search strategy: Beam Search



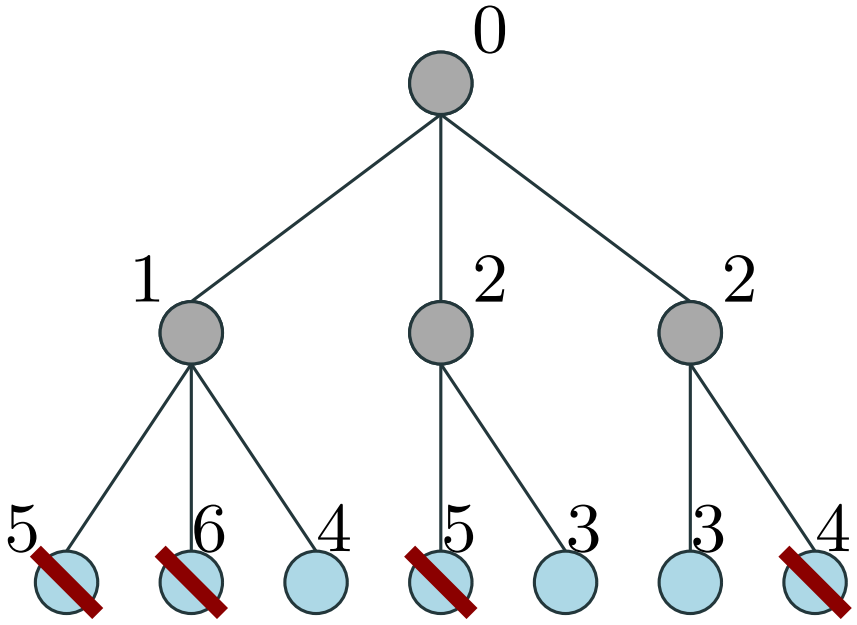
Another search strategy: Beam Search



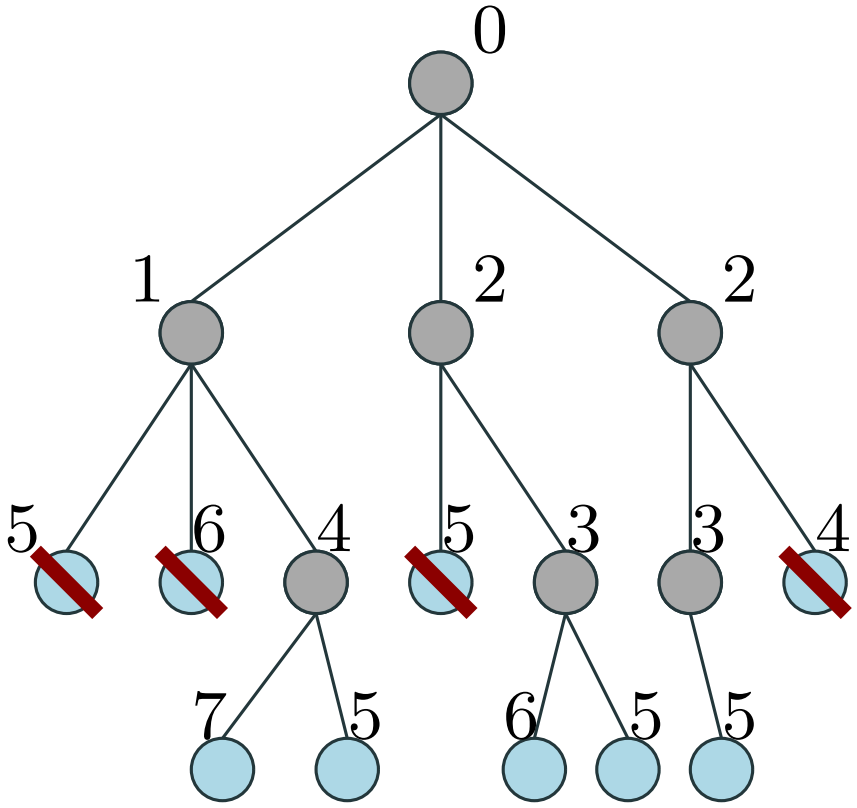
Another search strategy: Beam Search



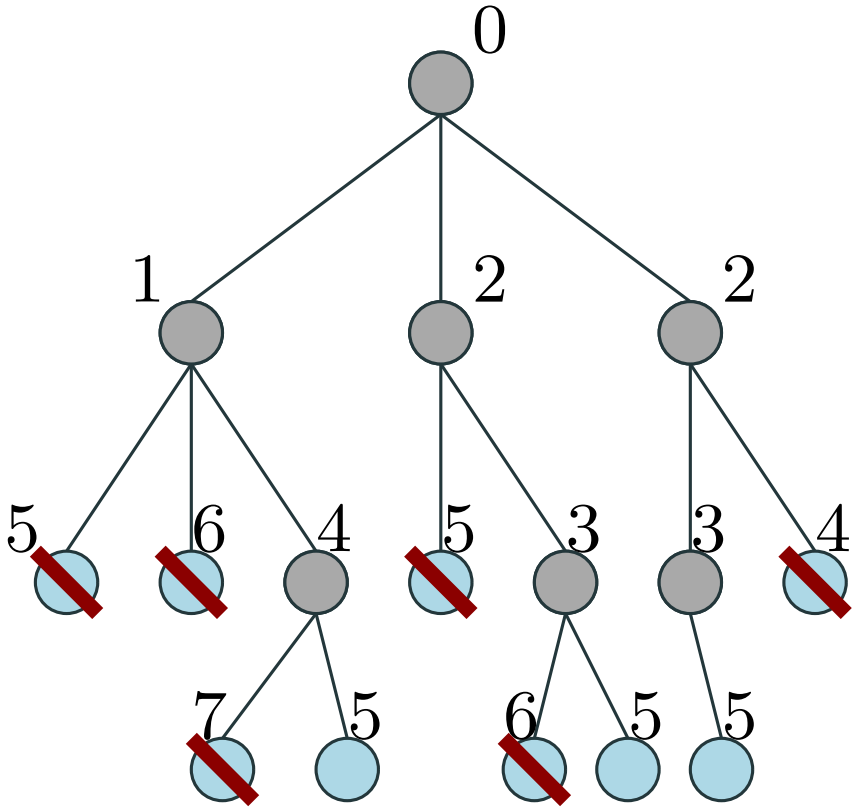
Another search strategy: Beam Search



Another search strategy: Beam Search



Another search strategy: Beam Search



Iterative Beam Search

- Runs a beam of size 1 (greedy)
- Then runs a beam of size 2, then 4, then 8 ...

Stops when no heuristic fathoming is done (proves optimality)

How to transform this branch-and-bound into an heuristic?

I. The search strategy
II. The guidance strategy

Guide functions: which node to choose?

which criterion?

the bound

the idle time

a bit of both $(\alpha \,.\, \text{bound} \;+\; (1-\alpha) \,.\, \text{idle})$

with weighted idle time (similar to the Liu & Reeves heuristic)

$$(\alpha \,.\, \text{bound} \;+\; (1-\alpha) \,.\, \text{weighted idle})$$

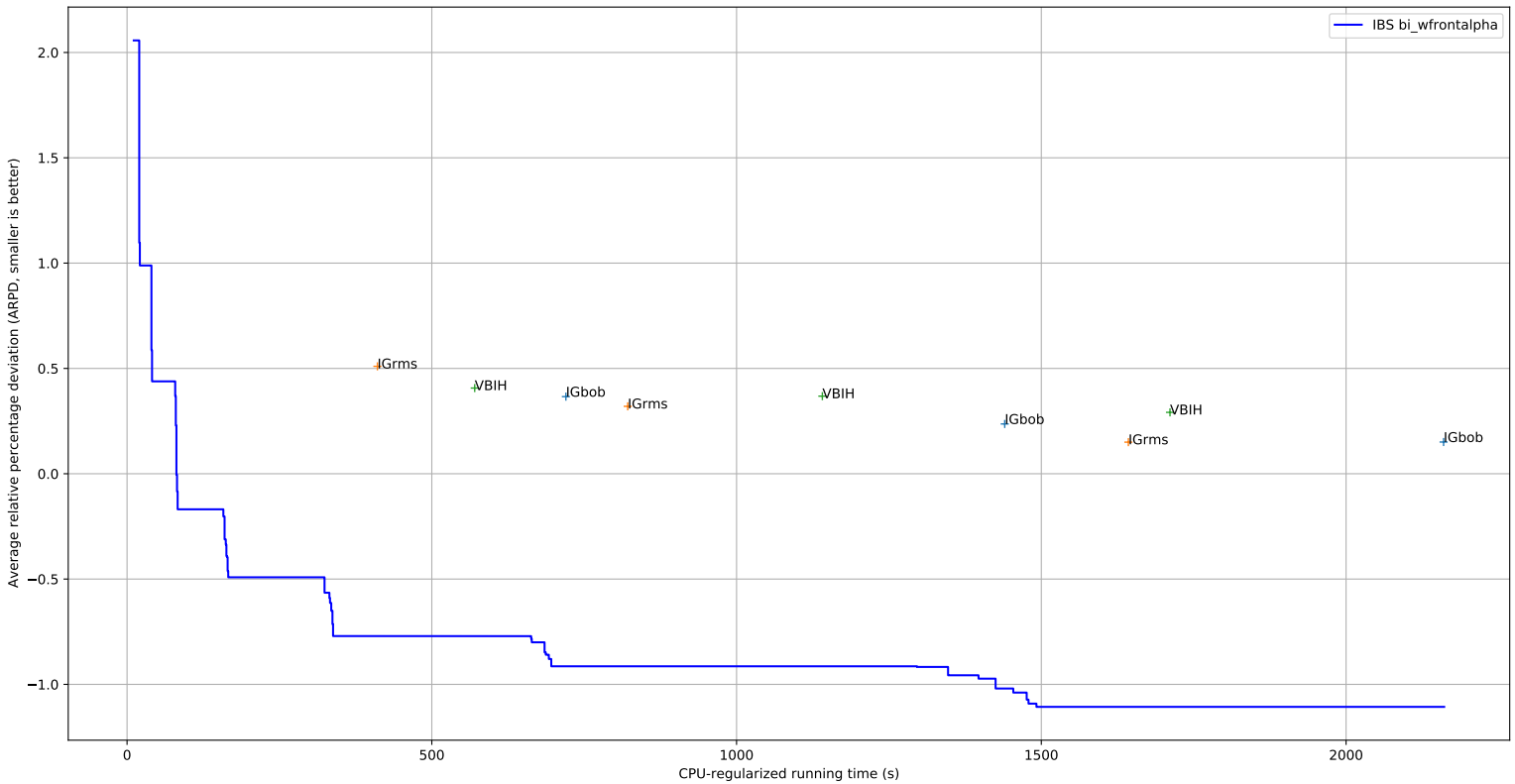
Results

Makespan variant

101/240 new-best-known solutions

- excellent on large instances (500+ jobs, 40+ machines)
- less efficient on “small” instances (100 jobs, 60 machines)

800 jobs, 60 machines

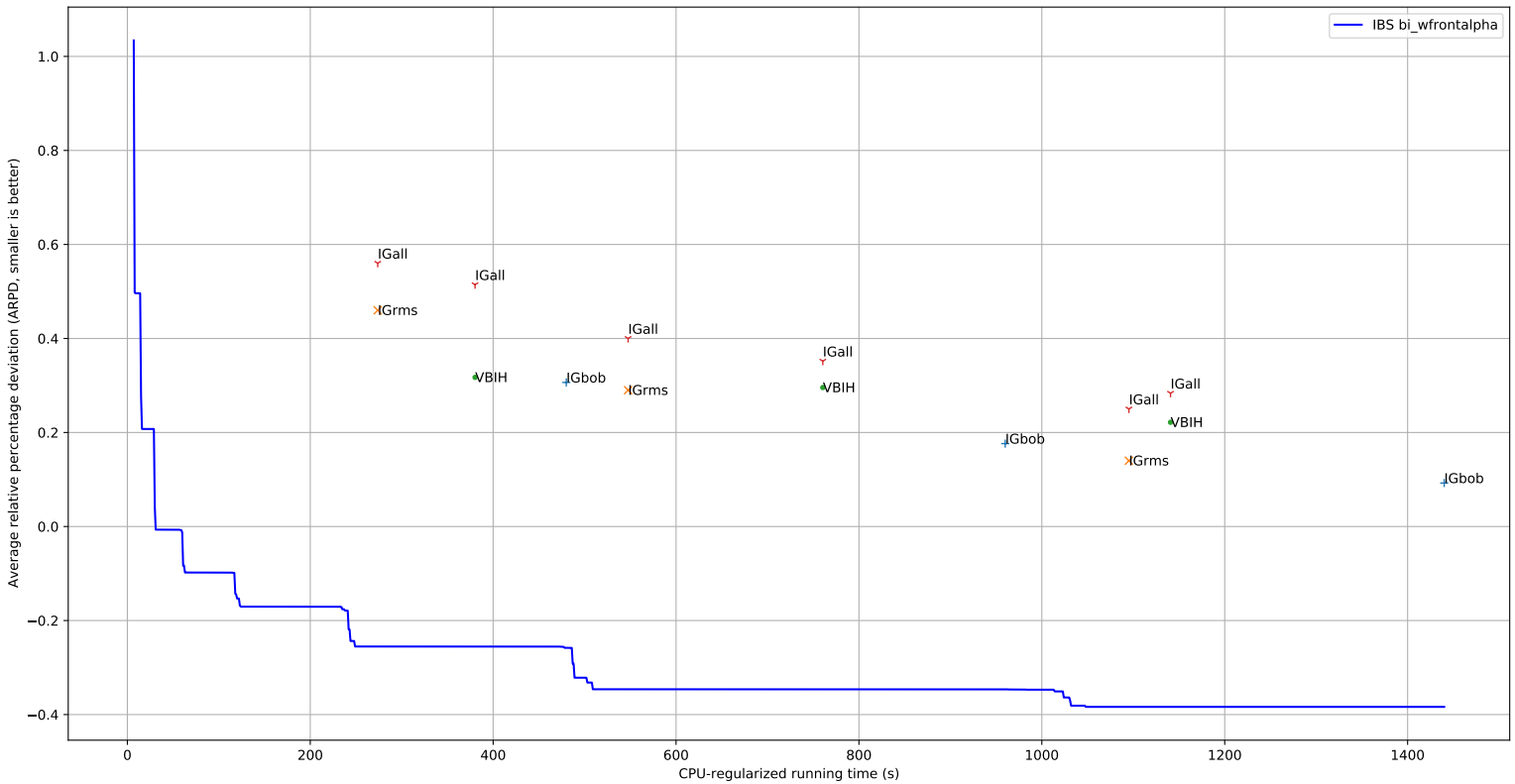


Makespan variant

101/240 new-best-known solutions

- excellent on large instances (500+ jobs, 40+ machines)
- less efficient on “small” instances (100 jobs, 60 machines)

800 jobs, 40 machines

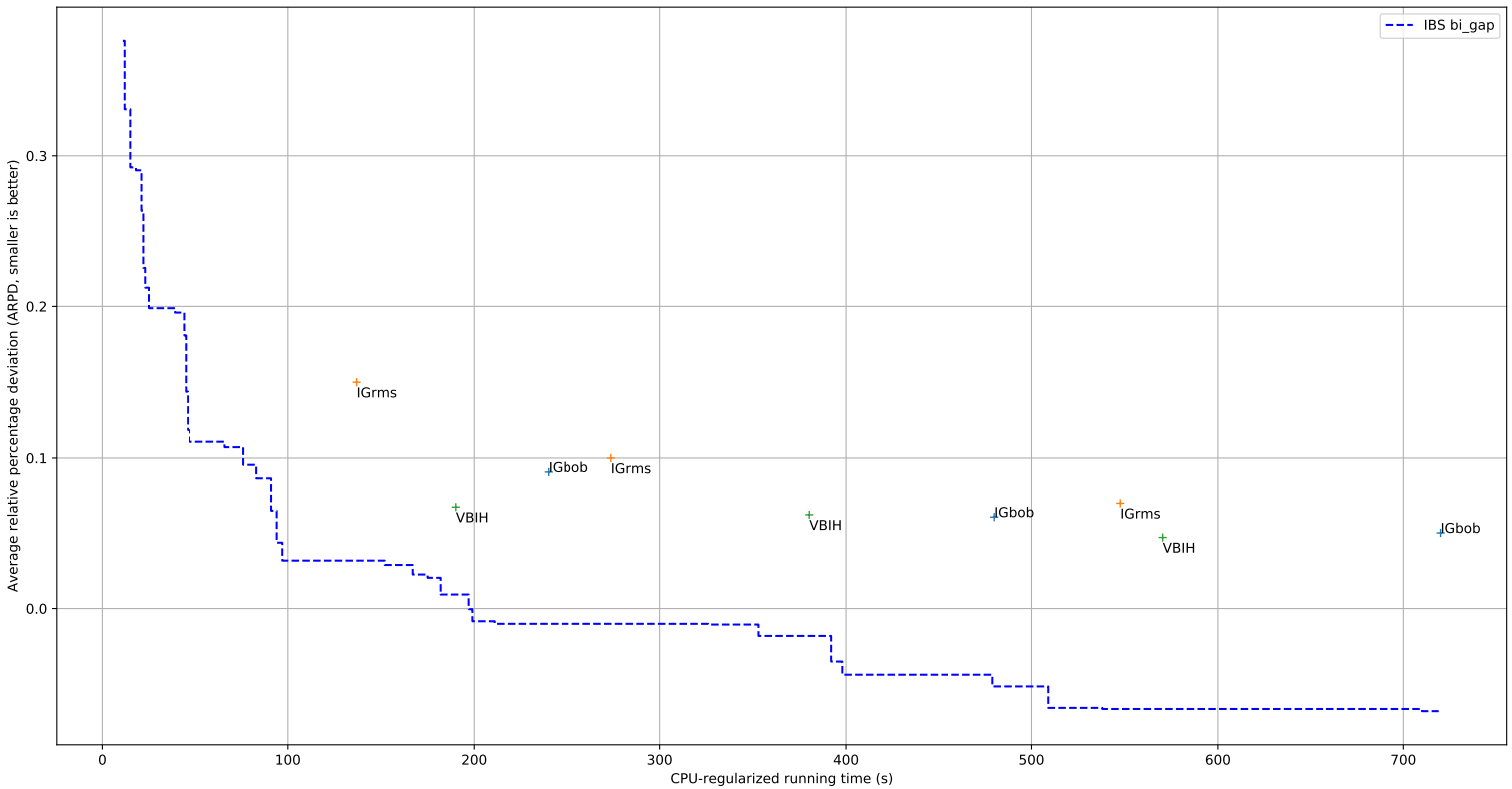


Makespan variant

101/240 new-best-known solutions

- excellent on large instances (500+ jobs, 40+ machines)
- less efficient on “small” instances (100 jobs, 60 machines)

800 jobs, 20 machines

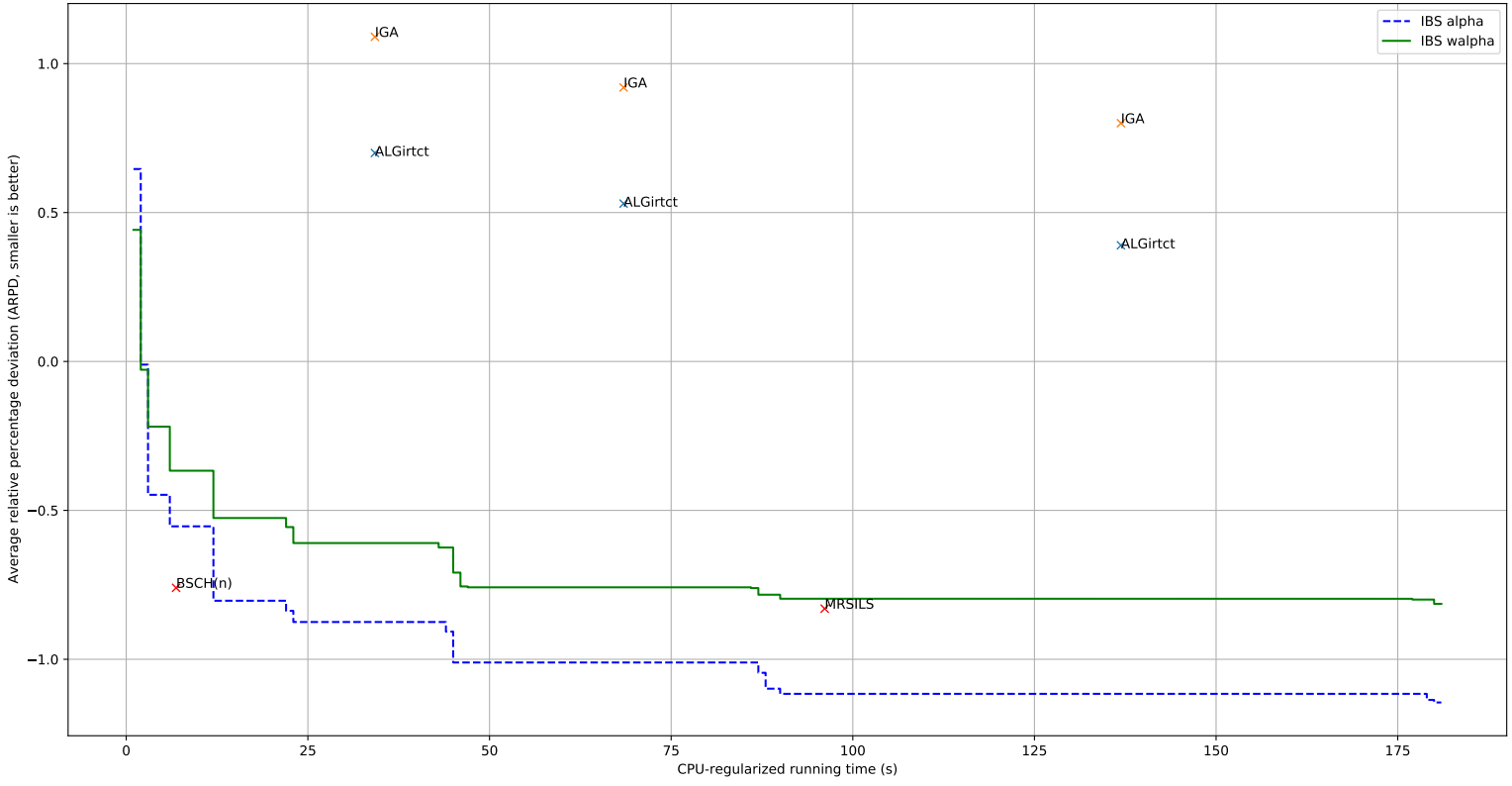


Total completion time variant

only the forward direction

51/120 new-best-known solutions

200 jobs, 20 machines

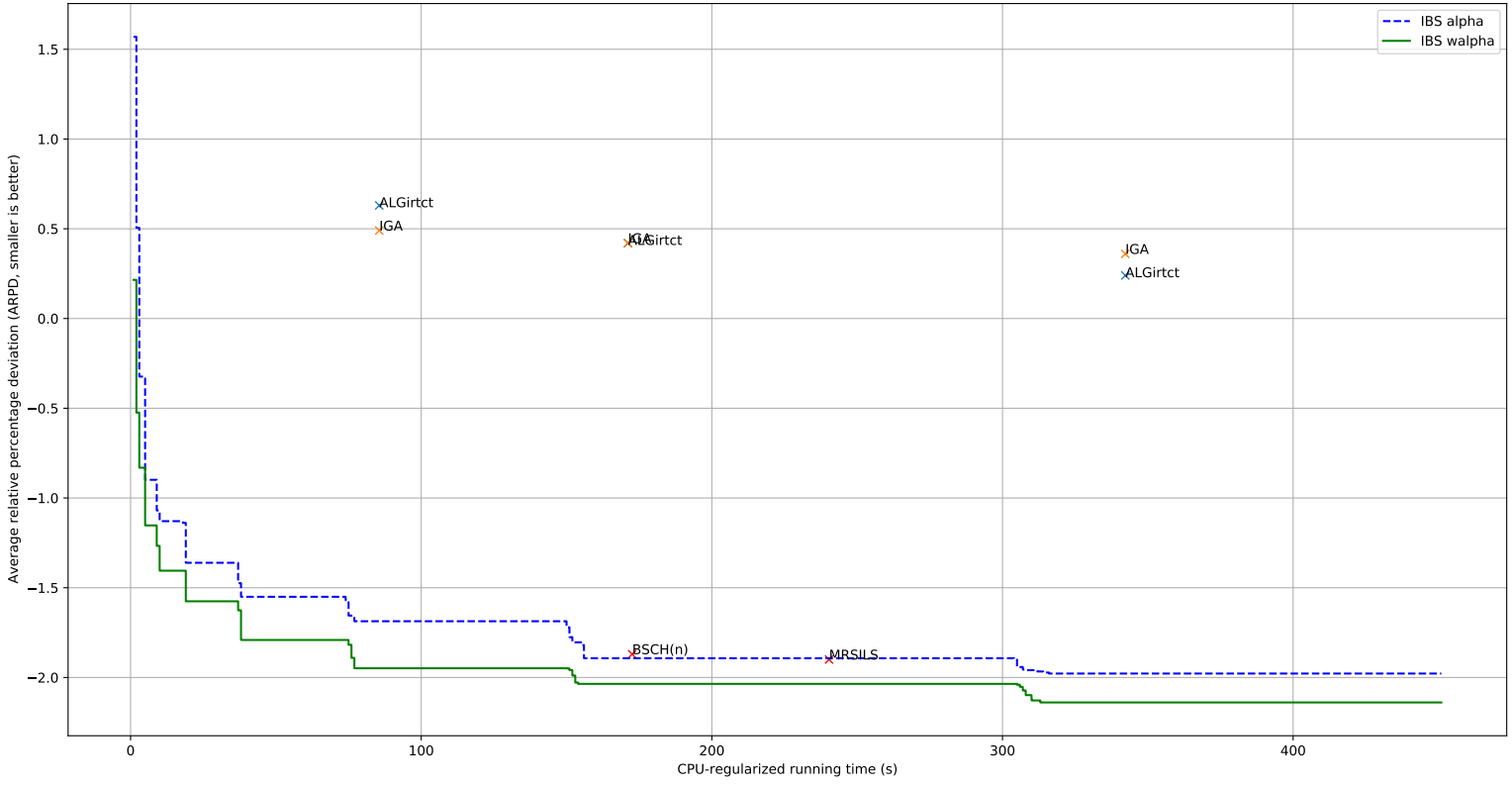


Total completion time variant

only the forward direction

51/120 new-best-known solutions

500 jobs, 20 machines



Conclusions

This methodology leads to interesting algorithms

sometimes, improving upon the state-of-the-art

used on other problems:

Conclusions

This methodology leads to interesting algorithms

sometimes, improving upon the state-of-the-art

used on other problems:

- **sequential ordering problem (ECAI2020, Q1)**
- **EURO/ROADEF 2018 challenge (EJOR, Q1)**
- general 2D cutting & packing problems
- longest common subsequence

- pre-print: *http://librallu.gitlab.io/pdfs/2020_pfsp_ibs.pdf*
- code: *https://github.com/librallu/dogs-pfsp*

FROM A BRANCH-AND-BOUND TO A STATE-OF-THE-ART HEURISTIC

An example on the permutation flowshop problem

Pablo Andres Focke[2], Vincent Jost[2], **Luc Libralesso**[1], Aurélien Secardin[2]

May, 6, 2021

1. LIMOS, Clermont-Ferrand, France
2. G-SCOP, Grenoble, France

email: **luc.libralesso@uca.fr**

References

Victor Fernandez-Viagas and Jose M Framinan. A best-of-breed iterated greedy for the permutation flowshop scheduling problem with makespan objective. *Computers & Operations Research*, 112: 104767, 2019.

Jan Gmys, Mohand Mezmaz, Nouredine Melab, and Daniel Tuyttens. A computationally efficient branch-and-bound algorithm for the permutation flow-shop scheduling problem. *European Journal of Operational Research*, 284(3):814–833, 2020.

Damla Kizilay, Mehmet Fatih Tasgetiren, Quan-Ke Pan, and Liang Gao. A variable block insertion heuristic for solving permutation flow shop scheduling problem with makespan criterion. *Algorithms*, 12 (5):100, 2019.

Federico Pagnozzi and Thomas Stützle. Automatic design of hybrid stochastic local search algorithms for permutation flowshop problems. *European journal of operational research*, 276(2): 409–421, 2019.

Caio Paziani Tomazella and Marcelo Seido Nagano. A comprehensive

review of branch-and-bound algorithms: Guidelines and directions for further research on the flowshop scheduling problem. *Expert Systems with Applications*, page 113556, 2020.