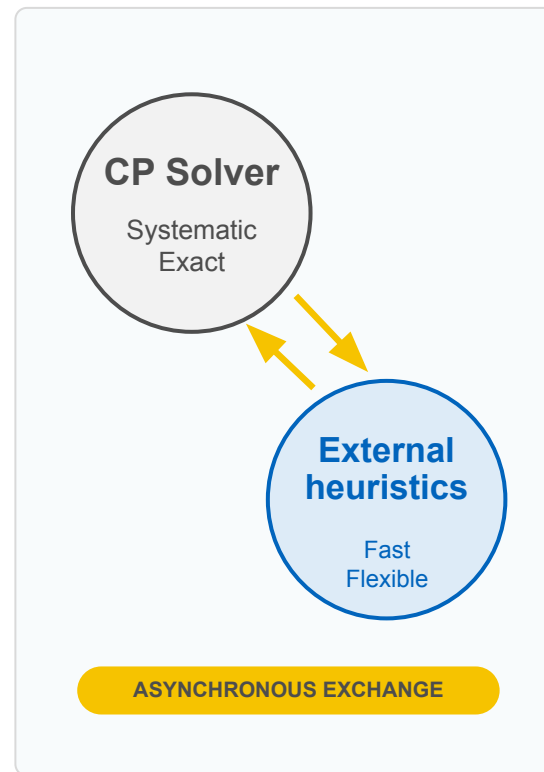


Accelerating Constraint Programming Solver with Parallel External Heuristics

Experiments on Scheduling and Routing Problems

Vilém Heinz · Šimon Zvára · Vít Knobloch · Zdeněk Hanzálek · Petr Vilím

Czech Technical University in Prague · CIIRC CTU · ScheduleOpt



Motivation: Different methods, complementary strengths

EXACT CP SOLVER

What it gives us

- Systematic, complete search
- Lower bounds and optimality proofs
- Can prove infeasibility

Where it struggles

- Large instances and short time limits
- Good incumbents may arrive late

(META)HEURISTICS

What they give us

- Problem-aware effective search
- Strong anytime behavior
- Scalable to large instances

Where they struggle

- No optimality guarantee
- Premature convergence and robustness

Combination: Best possible solution at any given time (with bounds and guarantees)

Goal: Strong anytime performance with exactness

1

Early search

Heuristics rapidly provide good feasible incumbents and promising search regions.

2

Advanced search

CP improves solutions, tightens bounds, and can ultimately prove optimality (infeasibility).

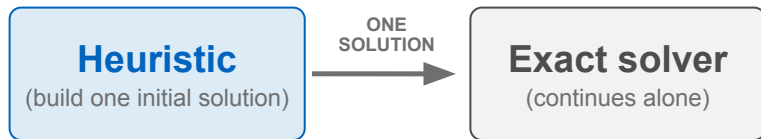
3

Throughout

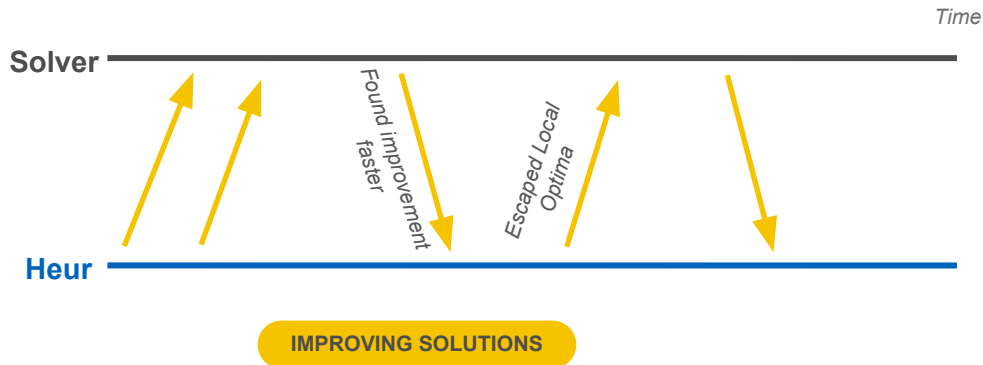
Workers can unstuck each other from local optima, increasing robustness across instances.

Continuous cooperation: from sequential to parallel

Typical sequential combination (warm start)



Collaborative parallel combination

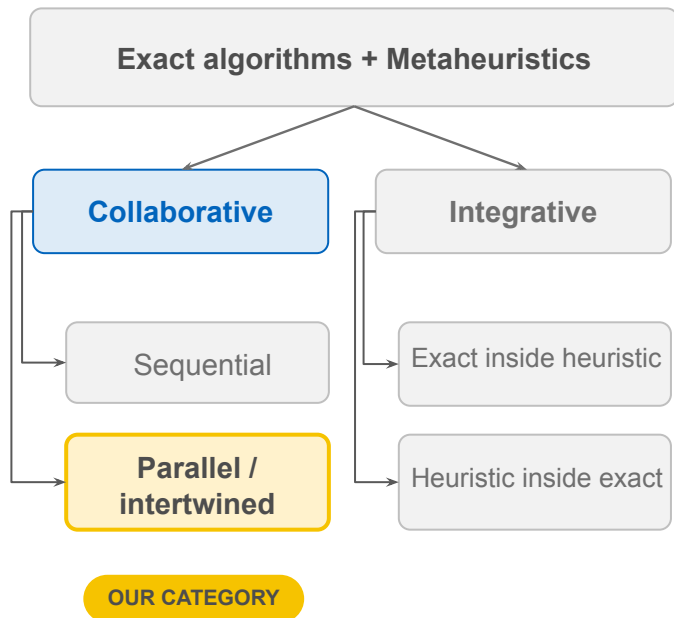


What changes

- Methods run at the same time
- Every new incumbent can be used immediately
- No restart and no fixed master/subroutine role

Positioning: collaborative, parallel, and general

Hybridization taxonomy



Closest collaborative parallel approaches

Approach	(i) Async	(ii) Exact	(iii) Black-box*	(iv) General
Talukdar et al. [71]	Yes	No	Yes ¹	No
Denzinger et al. [16]	No	Yes	No	No
Hung et al. [33]	Yes	Yes	No ²	No
This work	Yes	Yes	Yes	Yes

¹ Shared-memory communication. ² Cooperation depends on method-specific internals.

The closest existing works cover subsets of the attributes - none exhibits all four.

*Black-box here means external user-designed methods plugged into OptaCP; it is not a claim that the CP solver side can be changed for arbitrary one.

OptalCP: CP solver used

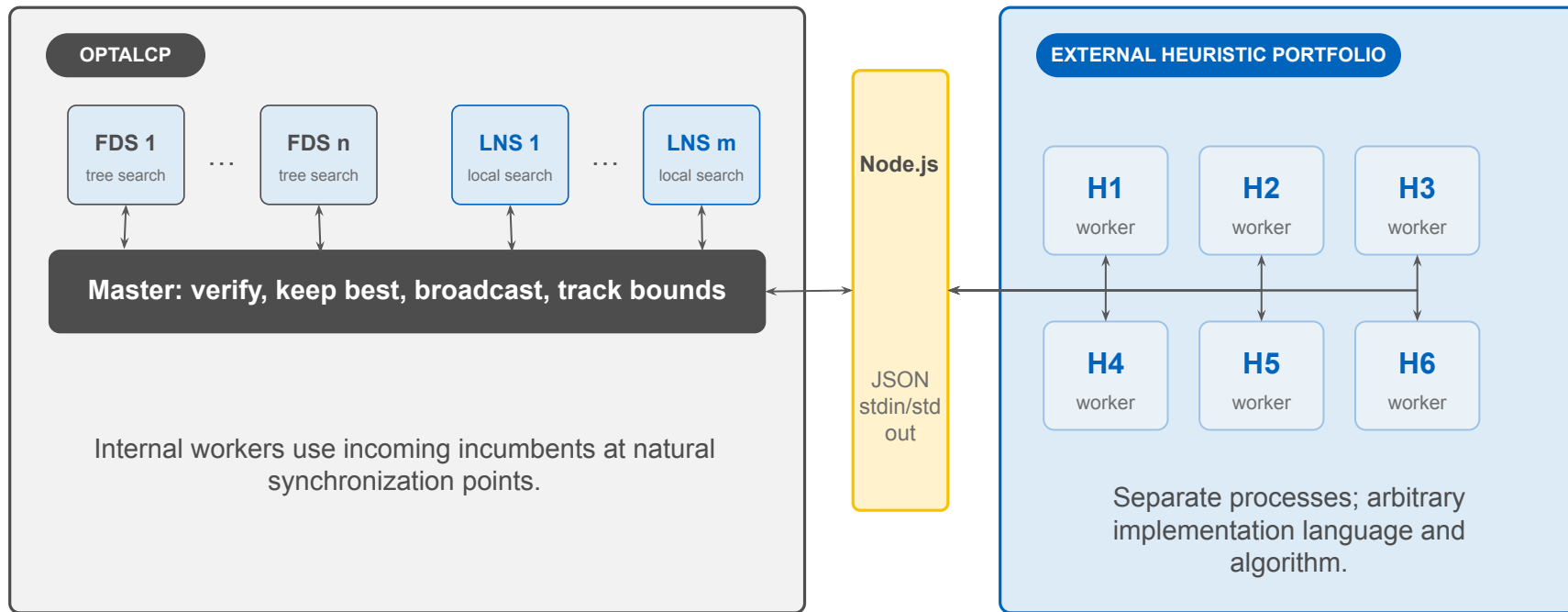
PROPERTIES

- Constraint Programming solver (similar to IBM CP Optimizer)
- Interval variables, sequences, cumulative resources...
- Written in C++20, APIs in TypeScript/JavaScript and Python.
- **Designed for parallel search.**
- **Async event-driven solving.**

PARALLEL COMPONENTS

- Failure Directed Search (FDS)
 - Systematic tree search
- Large Neighborhood Search (LNS)
 - Fast heuristic search
- Master
 - Verifies, broadcasts, tracks bounds

Hybrid architecture: Communication interface



Problem-specific pieces: the CP model, the metaheuristic representation, and translation between solution formats.

Code integration: Intentionally minimalistic

JAVASCRIPT GLUE

```
solver.onSolution = event => {  
  heuristic.stdin.write(  
    toHeuristicFormat(event.solution)  
  );  
};  
  
heuristic.stdout.on("data", data => {  
  solver.sendSolution(  
    fromHeuristicFormat(data)  
  );  
});
```

The user-written part only translates solution formats and forwards improvements.

Any language

- 1 A heuristic only needs to read and write the simple JSON solution format.

Asynchronous calls

- 2 The solver emits solutions through a callback; sendSolution injects an incumbent during search.

Coordination stays in the solver

- 3 Verification, distribution, best-value tracking, gaps, and termination are automatic.

Benchmark problems & configurations

FSSP

Flow-shop scheduling problem

60 largest Taillard instances
[69]

JSSP

Job-shop scheduling problem

100 mixed instances
[1, 68, 69, 15, 75]

RCPSP

Resource-constrained project scheduling problem

100 RG300 instances
[14]

TSP

Traveling salesman problem

84 TSPLIB instances
[59]

CVRP

Capacitated vehicle routing problem

27 CVRPLIB Set A instances
[72]

VRP-TW

Vehicle routing problem with time windows

56 Solomon (100 customers)
instances [67]

12-thread configurations

OptalCP

10L-2F

Reference

12CPO

Hybrid

Scheduling hybrid

5H-5L-2F

TSP / CVRP hybrid

2H-8L-2F

VRP-TW hybrid

3H-7L-2F

H = external heuristic worker
L = OptalCP LNS worker
F = OptalCP FDS worker
CPO = IBM CP Optimizer

CP Optimizer (12CPO) is a reference benchmark for OptalCP, not the focus of the study.

External (meta)heuristic portfolio

SCHEDULING

Same five metaheuristics for FSSP, JSSP, and RCPSP

GA

LNS

PSO

SA

TS

- Reusable algorithmic cores
- Problem-specific representation/operators

ROUTING

TSP

2-OPT local search + GA

CVRP

LS + stochastic-ranking GA

VRP-TW

adapted CVRP methods + memetic algorithm

Each heuristic gets one thread.
Population methods insert incoming solutions; local-search methods replace their current solution.

Experimental setup & comparison methods

120 s

time limit

12

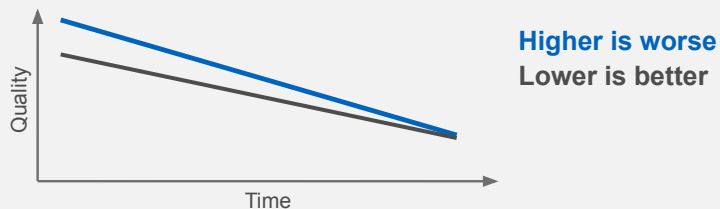
threads

3

random seeds

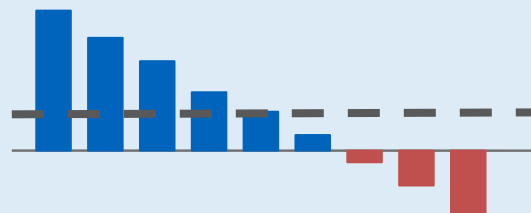
AMD 7700X, 64 GB RAM · Same CP model formulations
for OptalCP and CP Optimizer

Runtime Distribution Function contour



90th percentile of relative solution quality compared with the best solution observed in any run, any time by any method.

Per-instance relative difference (fixed time)

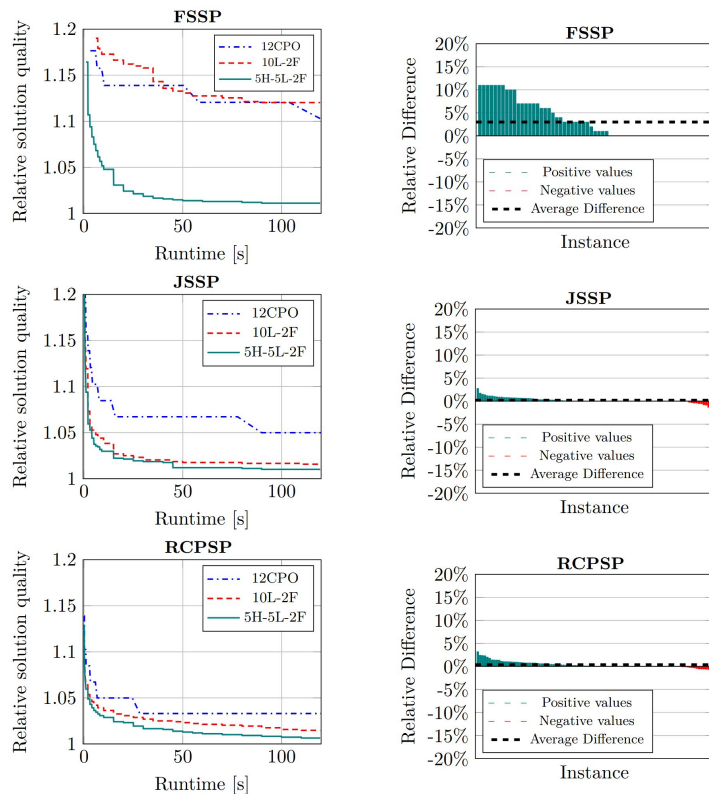


positive = hybrid better

negative = OptalCP better

Dashed horizontal line = average difference across instances.

Scheduling results



FSSP

+3%

Strong anytime improvement on almost every instance.

JSSP

marginal

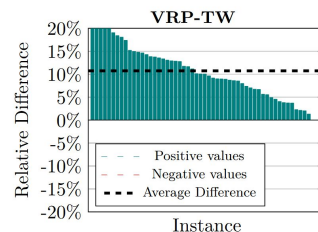
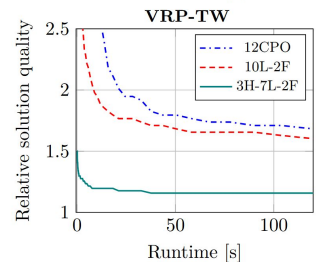
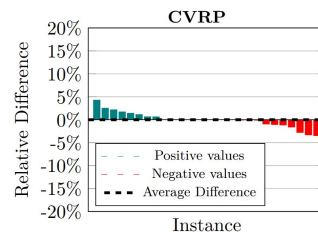
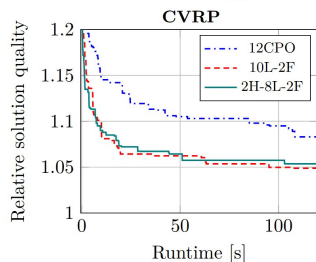
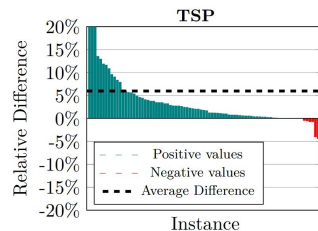
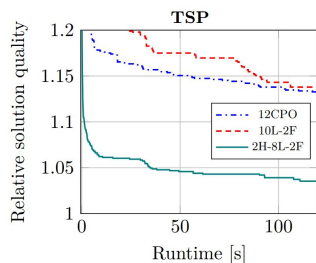
Hybrid and OptalCP track closely; solver alone is already strong.

RCPSP

marginal

Small average gain, with little separation in the anytime curves.

Routing results



TSP

+6%

Fast convergence and broad per-instance improvement.

CVRP

0%

No average or anytime improvement; performance varies by instance.

VRP-TW

+11%

Largest gain; the dedicated memetic heuristic is especially effective.

Robustness is valuable even when average gains are small

Across all six problem classes



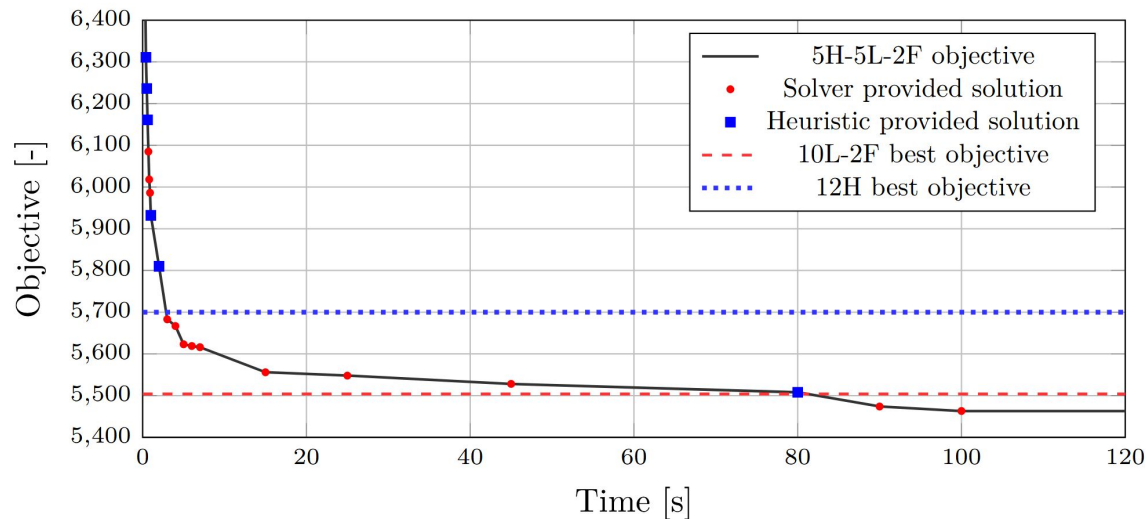
Hybrid solution quality versus OptalCP alone

Robustness in practise:
In 3 TSP instances, OptalCP alone found no feasible solution within 120s; heuristic solutions kept the hybrid search moving.

What robustness means here

- Different workers fail on different instances.
- A feasible heuristic solution can rescue a weak solver start.
- The solver can later improve heuristic solutions.
- **Bounds, gaps, and optimality proofs remain available even if solver does not provide solutions.**

Real-time cooperation example



Reading the graph

- Heuristic workers drive the rapid initial improvements.
- The CP solver becomes more influential later.
- A heuristic improves again around 80s after the search moves to a new region.
- The hybrid finishes below both solver-only and heuristic-only endpoints on this instance.

Illustrative instance only: a full comparison of communicating versus non-communication portfolio remains future work.

What the experiments (do not) show

SUPPORTED

- Hybrid configuration improves solution quality and convergence on 3 of 6 tested problems.
- Hybrid configuration improved robustness.
- Anecdotal evidence implies that communication can actively contribute to finding better solution.

OPEN

- Effect of the heuristic portfolio strength on the overall improvement.
- Statistically relevant comparison isolating the effect of real-time communication.
- *Analysis of the amount of contributed solutions by each method in the runtime (how much solutions provided justifies the method).*

Future case study: solver-only | heuristic-only | non-communicating portfolio | communicating hybrid (with significance tests).

Takeaways and next steps

1

Practical interface

External user-written methods communicate with OptaCP asynchronously through a minimal API.

2

Empirical benefit

Clear gains on FSSP, TSP, and VRP-TW; additional robustness on the remaining classes.

3

Exactness retained

The CP solver still supplies bounds, gaps, infeasibility detection, and optimality proofs.

Future work:

Extensive experiments

Richer information exchange

Adaptive worker selection

Learning-guided cooperation

Thank you! · Questions?

References

[16] J. Denzinger and T. Offermann. On cooperation between evolutionary algorithms and other search paradigms. CEC, 1999.

[33] Y.-F. Hung and W.-C. Chen. A heterogeneous cooperative parallel search of branch-and-bound and tabu search. Journal of Global Optimization, 2011.

[34] IBM Corporation. IBM ILOG CPLEX CP Optimizer, 2026.

[41] P. Laborie and D. Godard. Self-adapting large neighborhood search: Application to single-mode scheduling problems. MISTA, 2007.

[50] R. Pellerin, N. Perrier, and F. Berthaut. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. EJOR, 2020.

[53] P. Vilím and D. O. Fernández Pons. OptalCP: Local Constraint Programming Solver, 2024.

[56] J. Puchinger and G. R. Raidl. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. 2005.

[60] M. G. C. Resende and C. C. Ribeiro. Runtime distributions. In Optimization by GRASP, 2016.

[71] S. Talukdar, L. Baerentzen, A. Gove, and P. De Souza. Asynchronous teams: Cooperation schemes for autonomous agents. Journal of Heuristics, 1998.

[73] P. Vilím, P. Laborie, and P. Shaw. Failure-directed search for constraint-based scheduling. CPAIOR, 2015.